ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

DESAIN DAN ANALISIS ALGORITMA GENETIKA UNTUK MENGOPTIMASI ARSITEKTUR ALGORITMA JARINGAN SARAF TIRUAN DALAM PENGENALAN DIGIT YANG MENGALAMI DERAU PADA STUDI KASUS: SPOJ HIR HARD IMAGE RECOGNITION

YOLANDA HERTITA PRATAMA
NRP 05111640000052

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Ridho Rahman H., S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR - IF184802

DESAIN DAN ANALISIS ALGORITMA GENETIKA UNTUK MENGOPTIMASI ARSITEKTUR ALGORITMA JARINGAN SARAF TIRUAN DALAM PENGENALAN DIGIT YANG MENGALAMI DERAU PADA STUDI KASUS: SPOJ HIR HARD IMAGE RECOGNITION

YOLANDA HERTITA PRATAMA
NRP 05111640000052

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Ridho Rahman H., S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESES - IF184802

DESIGN AND ANALYSIS OF GENETIC ALGORITHM TO OPTIMIZE ARTIFICIAL NEURAL NETWORK ARCHITECTURE FOR NOISY DIGIT RECOGNITION IN CASE STUDY OF HARD IMAGE RECOGNITION FROM SPHERE ONLINE JUDGE

YOLANDA HERTITA PRATAMA
NRP 05111640000052

Supervisor 1
Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2
Ridho Rahman H., S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS ALGORITMA GENETIKA UNTUK MENGOPTIMASI ARSITEKTUR ALGORITMA JARINGAN SARAF TIRUAN DALAM PENGENALAN DIGIT YANG MENGALAMI DERAU PADA STUDI KASUS: SPOJ HIR HARD IMAGE RECOGNITION

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
Yolanda Hertita Pratama
NRP. 05111640000052

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom. .....
NIP. 197002131994021001 (Pembimbing 1)

Ridho Rahman H., S.Kom., M.Sc. .....
NIP. 198702132014041001 (Pembimbing 2)

SURABAYA
JANUARI 2020

[*Halaman ini sengaja dikosongkan*]

ABSTRAK

DESAIN DAN ANALISIS ALGORITMA GENETIKA UNTUK MENGOPTIMASI ARSITEKTUR ALGORITMA JARINGAN SARAF TIRUAN DALAM PENGENALAN DIGIT YANG MENGALAMI DERAU PADA STUDI KASUS: SPOJ HIR HARD IMAGE RECOGNITION

Nama	:	Yolanda Hertita Pratama
NRP	:	05111640000052
Departemen	:	Departemen Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	Ridho Rahman H., S.Kom., M.Sc.

Abstrak

Jaringan syaraf tiruan merupakan salah satu metode dalam mengolah informasi yang terinspirasi oleh sistem sel syaraf biologis. Pada dasarnya jaringan syaraf tiruan adalah sejumlah satuan masukan dan luaran yang saling terkoneksi yang setiap koneksinya memiliki bobot tersendiri. Pada proses pembelajaran, jaringan syaraf belajar dengan mengubah bobot sehingga pada akhirnya dapat memprediksi kelas target yang tepat dari masukan yang telah diberikan. Hingga kini, penyusunan arsitektur dari jaringan syaraf tiruan masih merupakan salah satu topik yang menarik dalam riset. Beberapa algoritma optimasi untuk menentukan arsitektur yang optimal telah digunakan, salah satunya adalah algoritma genetika. Algoritma genetika adalah teknik pencarian dalam bidang komputasi untuk menemukan solusi benar atau pendekatan untuk masalah optimasi dan pencarian. Teknik dalam GA didasarkan pada biologi

evolusioner seperti pewarisan, mutasi, seleksi dan crossover. Pada tugas akhir ini, penulis akan merancang penyelesaian permasalahan pengenalan digit pada studi kasus permasalahan SPOJ Hard Image Recognition dengan mengimplementasikan jaringan syaraf tiruan yang telah dioptimasi menggunakan algoritma genetika. Penyusunan algoritma dan pembangunan dataset yang tepat juga dibutuhkan untuk mengatasi permasalahan derau yang terdapat pada gambar. Perancangan arsitektur yang dibuat berhasil mendapatkan nilai tertinggi sebesar 108 pada SPOJ Hard Image Recognition.

Kata Kunci: jaringan syaraf tiruan; algoritma genetika; optimasi; pengenalan digit

ABSTRACT

DESIGN AND ANALYSIS OF GENETIC ALGORITHM TO OPTIMIZE ARTIFICIAL NEURAL NETWORK ARCHITECTURE FOR NOISY DIGIT RECOGNITION IN CASE STUDY OF HARD IMAGE RECOGNITION FROM SPHERE ONLINE JUDGE

Name	:	Yolanda Hertita Pratama
Student ID	:	05111640000052
Department	:	Informatics Department, Faculty of Intelligent Electrical and Informatics Technology, ITS
Supervisor I	:	Rully Soelaiman, S.Kom., M.Kom.
Supervisor II	:	Ridho Rahman H., S.Kom., M.Sc.

Abstract

Artificial neural network is a technique for information processing inspired by biological neural networks. Basically, a network represents group of input nodes and group of output nodes that are connected to each other that each connection have a weight. On training process, neural networks learn by updating these weight values to make the networks give correct prediction for various input data. The development of an artificial neural network architecture have been a hot topic of research for many years. Some of optimization algorithm are used to find the best architecture, one of them is genetic algorithm. Genetic algorithm is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithm use evolutionary techniques such as inheritance, mutation, selection, and crossover. At this thesis, the author will design a solution to the digit recognition problem on a case study of SPOJ Hard Image Recognition by

implementing artificial neural network optimized by genetic algorithm. The algorithm development and the creation of dataset are used to get over with noises on image. The proposed architecture development succesfully achieved a high score of 108 at SPOJ Hard Image Recognition.

Keywords: artificial neural network; genetic algorithm; optimization; digit recognition

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa. Atas rahmat dan kasih sayangNya, penulis dapat menyelesaikan tugas akhir dan laporan akhir dalam bentuk buku ini.

Pengerjaan buku ini penulis tujukan untuk mengeksplorasi lebih mendalam topik-topik yang tidak diwadahi oleh kampus, namun banyak menarik perhatian penulis. Selain itu besar harapan penulis bahwa pengerjaan tugas akhir sekaligus pengerjaan buku ini dapat menjadi batu loncatan penulis dalam menimba ilmu yang bermanfaat.

Penulis ingin menyampaikan rasa terima kasih kepada banyak pihak yang telah membimbing, menemani dan membantu penulis selama masa pengerjaan tugas akhir maupun masa studi.

1. Kedua orangtua penulis, serta keluarga penulis sebagai penyemangat pertama yang telah memberikan dukungan doa dan moral sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Bapak Rully Soelaiman S.Kom., M.Kom., selaku pembimbing penulis yang telah memberikan dukungan baik berupa bimbingan, ilmu, semangat, perhatian dan nasihat selama masa studi penulis.
3. Bapak Ridho Rahman H., S.Kom., M.Sc., selaku pembimbing penulis yang telah memberikan dukungan baik berupa bimbingan dan ilmu dalam mengerjakan Tugas Akhir.
4. Seluruh dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
5. Fata Nugraha yang selalu memberikan semangat dan nasehat yang baik kepada penulis.
6. Teman-teman admin laboratorium Komputasi Cerdas dan Vi-

si yang telah menemani dan membantu penulis dalam mengerjakan tugas kuliah dan yang lain.
7. Nurlita, Nirmala, dan Ferdinand yang telah menemani dan membantu penulis selama menempuh masa studi di Departemen Informatika ITS.
8. Dandy dan Fadli selaku teman tim Data Mining penulis, yang telah memberi banyak pengalaman.
9. Teman-teman mahasiswa Departemen Informatika ITS yang senantiasa membantu, menemani, memberi semangat dan kenangan selama kurang lebih 3,5 tahun masa studi.
10. Serta semua pihak yang turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa buku ini jauh dari kata sempurna. Maka dari itu, penulis memohon maaf apabila terdapat salah kata maupun makna pada buku ini. Penulis berharap buku ini dapat berkontribusi dalam ilmu pengetahuan dan memberikan manfaat bagi pembaca.

Surabaya, Januari 2020

Yolanda Hertita Pratama

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR PSEUDOCODE

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi pengerjaan, dan sistematika penulisan Tugas Akhir.

1.1. Latar Belakang

Citra merupakan salah satu komponen yang sangat penting dalam informasi visual. Manusia berlomba-lomba untuk mengembangkan algoritma pengenalan citra yang cepat dan akurat. Salah satu bagian dari pengenalan citra adalah pengenalan karakter yang berguna dalam aspek kehidupan manusia seperti untuk pengenalan karakter dari plat nomor kendaraan, pengenalan karakter dari nomor rumah, membaca tulisan tangan, dan lain-lain.

Persoalan yang akan diselesaikan pada tugas akhir ini mengacu pada persoalan pada *Sphere Online Judge* dengan judul *High Image Recognition* (HIR) [1]. Pada permasalahan ini terdapat sekumpulan karakter 'X' dan '.' yang merepresentasikan citra biner dari 6 angka. 'X' menyatakan *foreground* dan '.' menyatakan *background* dari citra tersebut. Penyelesaian yang diharapkan adalah mendapatkan keluaran yang berupa 6 buah angka yang sesuai dengan citra melalui proses pengenalan karakter. Tantangan pada permasalahan ini adalah terdapat citra yang memiliki derau (*noise*) sehingga sulit untuk dikenali. Oleh karena itu dibutuhkan desain algoritma yang tepat untuk mengatasi permasalahan ini. Algoritma yang akan digunakan untuk menyelesaikan permasalahan ini adalah algoritma jaringan syaraf tiruan.

Jaringan syaraf tiruan memerlukan waktu pelatihan dan kebutuhan sejumlah parameter yang perlu ditentukan. Kelemahan dari jaringan syaraf yiruan adalah tingkat pemahaman atau *insight* yang rendah, karena pemahaman di balik bobot pembelajaran yang di-

dapatkan dari jaringan syaraf tiruan sulit untuk diinterpretasikan. Namun dibalik kelemahan tersebut, jaringan syaraf yiruan memiliki kelebihan, yaitu jaringan syaraf tiruan memiliki toleransi yang besar terhadap data derau (*noise*). Selain itu jaringan syaraf tiruan kokoh dalam hal kesalahan dalam pembelajaran data. Pada tugas akhir ini parameter jaringan syaraf tiruan akan dioptimasi dengan menggunakan algoritma genetika sehingga dapat menghasilkan arsitektur dan bobot yang optimal. Hasil dari tugas akhir ini diharapkan dapat memberikan gambaran mengenai algoritma yang digunakan untuk menyelesaikan permasalahan secara optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut :

1. Bagaimana cara mendeteksi area yang merupakan karakter dan menghilangkan derau pada gambar?
2. Bagaimana cara memprediksi karakter yang tepat sebagai luaran dari gambar dengan menggunakan algoritma Jaringan Saraf Tiruan?
3. Bagaimana cara mengoptimasi parameter pada algoritma Jaringan Saraf Tiruan dengan menggunakan algoritma Genetika?

1.3. Batasan Masalah

Permasalahan yang dibahas pada tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Dataset yang digunakan menggunakan dataset yang dibuat secara otomatis menggunakan program yang dibangun.

2. Implementasi algoritma menggunakan bahasa pemrograman Python dan C++.

Berikut merupakan batasan pada situs *Sphere Online Judge* :

1. Batas maksimum uji kasus (test case) sebanyak 250.
2. Batas maksimum waktu eksekusi program adalah 0.1 s.
3. Batas maksimum memori yang diberikan adalah 1535 MB.
4. Batas maksimum kode sumber yang diberikan adalah 0.15 MB.
5. Batas minimum tinggi dan lebar citra masukan adalah 10.
6. Batas maksimum tinggi dan lebar citra masukan adalah 250.

1.4. Tujuan

Tujuan tugas akhir ini adalah sebagai berikut:

1. Melakukan analisis dan mendesain algoritma untuk melakukan segmentasi karakter dan menghilangkan derau pada citra SPOJ *Hard Image Recognition* (HIR).
2. Membangun dataset yang tepat untuk pelatihan algoritma Jaringan Saraf Tiruan.
3. Menentukan desain algoritma Jaringan Saraf Tiruan dengan optimasi algoritma Genetika untuk mengatasi permasalahan pengenalan citra yang optimal pada SPOJ *Hard Image Recognition* (HIR).

1.5. Manfaat

Manfaat tugas akhir ini adalah sebagai berikut:

1. Membantu memahami penggunaan algoritma yang tepat untuk melakukan segmentasi karakter pada citra SPOJ *Hard Image Recognition* (HIR).
2. Membantu memahami penggunaan algoritma yang tepat untuk mengatasi dan menghilangkan derau pada citra SPOJ *Ha-*

rd Image Recognition (HIR).
3. Mengetahui metode pengenalan citra yang optimal dengan batasan-batasan yang disediakan.
4. Mengetahui hasil evaluasi kinerja optimasi dengan menggunakan algoritma Genetika pada algoritma Jaringan Saraf Tiruan.

1.6. Metodologi

Metodologi pengerjaan yang digunakan pada tugas akhir ini memiliki beberapa tahapan. Tahapan-tahapan tersebut yaitu:

1. Penyusunan proposal
   Pada tahapan ini penulis memberikan penjelasan mengenai apa yang penulis akan lakukan dan mengapa Tugas Akhir ini dilakukan. Penjelasan tersebut dituliskan dalam bentuk proposal Tugas Akhir.
2. Studi literatur
   Pada tahapan ini penulis mengumpulkan referensi yang diperlukan guna mendukung pengerjaan Tugas Akhir. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel internet, atau sumber lain yang bisa dipertanggungjawabkan.
3. Desain
   Pada tahapan ini, penulis melakukan desain dataset yang akan digunakan untuk proses pembelajaran dan desain rancangan algoritma yang akan dilakukan untuk proses pengenalan karakter.
4. Implementasi algoritma
   Pada tahapan ini penulis mulai mengembangkan algoritma sesuai dengan hasil rancangan desain untuk menyelesaikan permasalahan. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman Python dan C++.
5. Pengujian dan evaluasi

Pada tahapan ini penulis menguji performa algoritma yang telah dibangun menggunakan dataset SPOJ *Hard Image Recognition* (HIR) pada sistem penilaian daring *Sphere Online Judge* dengan waktu eksekusi program yang harus kurang dari 0,1 detik, memori yang digunakan program saat dijalankan kurang dari 1535 MB, dan ukuran kode sumber yang dikirim kurang dari 0,15 MB.

6. Penyusunan buku
   Pada tahapan ini penulis menyusun hasil pengerjaan Tugas Akhir mengikuti format penulisan Tugas Akhir.

1.7. Sistematika Penulisan

Sistematika laporan Tugas Akhir yang akan digunakan adalah sebagai berikut :

1. BABI : PENDAHULUAN
   Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.
2. BAB II : DASAR TEORI
   Bab ini berisi dasar teori mengenai permasalahan dan algoritma penyelesaian yang digunakan dalam Tugas Akhir
3. BAB III : DESAIN
   Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.
4. BAB IV : IMPLEMENTASI
   Bab ini berisi implementasi berdasarkan desain algortima yang telah dilakukan pada tahap desain.
5. BAB V : PENGUJIAN DAN EVALUASI
   Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.
6. BAB VI : PENUTUP
   Bab ini berisi kesimpulan dan saran yang didapat dari hasil

uji coba yang telah dilakukan.

BAB II
DASAR TEORI

Pada bab ini dijelaskan deskripsi permasalahan yang dikerjakan kemudian dilanjutkan dengan menjelaskan beberapa dasar teori yang akan penulis gunakan sebagai landasan pengerjaan tugas akhir ini.

2.1. Deskripsi Umum Permasalahan

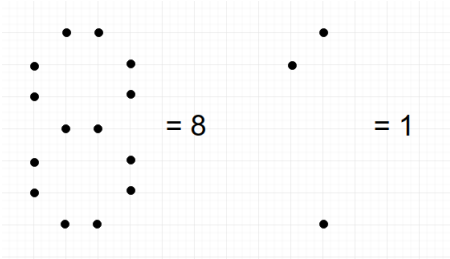
Pada sub bab berikut ini akan dijelaskan mengenai gambaran masalah secara umum, landasan teori yang digunakan, serta strategi penyelesaian yang digunakan untuk menyelesaikan permasalahan dalam Tugas Akhir ini.

2.1.1. Gambaran Masalah dari Penyelesaian secara Naive

Diberikan *Array* berukuran H x W yang merupakan representasi dari citra yang berisi digit-digit didalamnya. Tujuan dari permasalahan ini adalah untuk mengenali digit-digit yang terdapat pada citra tersebut. Batasan dari permasalahan tersebut adalah waktu eksekusi sehingga menghasilkan keluaran yang harus cepat serta algoritma yang digunakan tidak boleh melebihi batas panjang yang ditentukan. Gambar 2.1 merupakan contoh citra dari permasalahan yang akan diselesaikan.

Sistem harus dapat mengenali citra 2.1 sebagai digit ”371981”. Pendekatan umum sekaligus *naive* yang dilakukan untuk menyelesaikan permasalahan ini adalah melakukan segmentasi pada citra tersebut untuk mendapatkan bagian berwarna hitam yang

Gambar 2.1 Contoh Citra Permasalahan Pengenalan Digit



Gambar 2.2 Penyelesaian Menggunakan Titik Sudut Digit

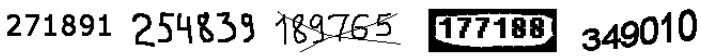
merupakan *foreground* atau bagian yang merupakan digit kemudian melakukan menghitung posisi bagian yang menyala yang dianggap merupakan sudut pada digit tersebut.

Pada Gambar 2.2 merupakan contoh untuk penyelesaian menggunakan titik sudut digit. Apabila posisi titik yang menyala seperti pada gambar sebelah kiri, maka digit tersebut dikenali sebagai digit 8. Apabila posisi titik yang menyala seperti pada gambar sebelah kanan, maka digit tersebut dikenali sebagai digit 1. Begitu pula untuk setiap digit yang lain dari 0 sampai dengan 9 akan disimpan kunci posisi digit yang menyala untuk menentukan digit yang sesuai.

2.1.2. Masalah Akibat Penyelesaian secara Naive

Permasalahan dapat berkembang seperti ukuran digit yang bervariasi, jenis tipografi digit yang bervariasi, dan augmentasi yang terjadi dari digit tersebut seperti rotasi.

Strategi penyelesaian permasalahan secara *naive* akan memiliki kelemahan dalam mengatasi perkembangan permasalahan. Keterbatasan banyaknya kode sumber algoritma yang dipakai untuk menyelesaikan permasalahan dan batas waktu eksekusi untuk menghasilkan keluaran yang diharapkan merupakan alasan utama mengapa penyelesaian secara *naive* bukanlah penyelesaian yang di-



Gambar 2.3 Contoh Citra Masukan BMP

harapkan. Sebagai contoh, untuk digit 1 saja memiliki banyak variasi dalam penulisan. Digit 1 dapat memiliki perbedaan dalam tipografi tulisan tangan dan menggunakan tipografi mesin.

Sistem diharapkan untuk menghasilkan luaran seakurat dan seefektif mungkin. Banyaknya variasi dalam penulisan digit ini menyebabkan pendekatan penyelesaian secara *naive* menjadi kurang tepat dan dibutuhkannya pendekatan lain untuk menyelesaikan permasalahan ini.

2.2. Deskripsi Permasalahan HIR

Permasalahan yang dikerjakan pada tugas akhir ini diangkat dari permasalahan pada SPOJ *Hard Image Recognition* (HIR). Pada permasalahan ini terdapat sekumpulan karakter 'X' dan '.' yang merepresentasikan citra biner dari 6 angka. 'X' menyatakan *foreground* dan '.' menyatakan *background* dari citra tersebut. Citra tersebut memiliki berbagai macam tipe tulisan dan beberapa mengalami transformasi. Terdapat pula citra yang memiliki derau, seperti derau coretan. Program yang dibuat harus dapat mengenali 6 digit sesuai dengan citra tersebut. Permasalahan ini memiliki tipe *challenge*, dimana penilaian pada SPOJ akan dilakukan berdasarkan nilai. Semakin banyak program dapat mengenali citra dengan benar, maka semakin tinggi nilai yang didapat. Pada situs penilaian daring SPOJ *Hard Image Recognition* (HIR) [1], terdapat 10 contoh citra masukan yang berupa berkas BMP seperti pada Gambar 2.3. Sedangkan pada situs forum diskusi kontes terbuka SPOJ *Hard Image Recognition* (HIR) [2], terdapat 13 contoh citra masukan yang berupa berkas TXT, yang berisi kumpulan karakter 'X' dan '.' sesuai dengan masukan pada soal seperti pada Gambar 2.4.

Gambar 2.4 Contoh Citra Masukan TXT

2.2.1. Parameter Masukan

Parameter masukan pada permasalahan SPOJ *Hard Image Recognition* (HIR) adalah sebagai berikut:

1. t, yaitu bilangan bulat yang menyatakan banyaknya jumlah kasus coba (*test case*), $t \leq 250$. Pada masing-masing kasus coba terdapat masukan H dan W.
2. H, yaitu bilangan bulat yang menyatakan tinggi dari citra masukan, $10 \leq H \leq 200$.
3. W, yaitu bilangan bulat yang menyatakan lebar dari citra masukan, $10 \leq W \leq 200$.

2.2.2. Batasan Permasalahan

Batasan pada permasalahan SPOJ *Hard Image Recognition* (HIR) adalah sebagai berikut:

1. Batas *runtime* atau waktu eksekusi program selama 0,1 detik.
2. Batas penggunaan memori sebesar 1536 MB.
3. Batas ukuran kode sumber yang dikirim sebesar 0,15 MB.

2.2.3. Keluaran Permasalahan

Hasil keluaran program adalah 6 digit yang merupakan hasil prediksi pada citra masukan pada masing-masing kasus coba.

2.3. Strategi Penyelesaian Permasalahan HIR

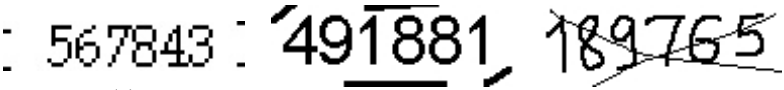
Pada bagian ini dibagi menjadi beberapa bagian subbab yang menjelaskan strategi penyelesaian permasalahan SPOJ *Hard Image Recognition* (HIR) mulai dari tahap praproses citra masukan, bagaimana menggunakan Jaringan Saraf Tiruan (*Artificial Neural Network*) sebagai algoritma utama dalam menyelesaikan permasalahan beserta persiapan dataset yang digunakan dalam pelatihan JST, dan bagaimana mengoptimasi arsitektur dari Jaringan Saraf Tiruan tersebut menggunakan algoritma Genetika (*Genetic Algorithm*).

2.3.1. Praproses Citra Digit

Pada permasalahan SPOJ *Hard Image Recognition* (HIR), terdapat sekumpulan karakter 'X' dan '.' yang merepresentasikan citra biner dari 6 digit. Tahapan pertama yang dilakukan adalah tahap praproses, tujuan dari tahapan ini antara lain untuk mendapatkan digit dengan melakukan segmentasi, menghilangkan derau pada digit, dan menormalisasi ukuran dari digit agar memiliki banyak fitur yang sama.

2.3.1.1. Praproses Citra Digit yang Mengalami Derau

Pada permasalahan SPOJ *Hard Image Recognition* (HIR) terdapat beberapa citra yang memiliki derau (*noise*) seperti derau titik atau *spot*, derau garis atau *border*, dan derau coretan atau *struckout*. Contoh dari citra yang mengalami permasalahan ini dapat dilihat pada Gambar 2.5



(a) (b) (c)

Gambar 2.5 Citra dengan Derau Titik (a), Citra dengan Derau Garis (b), dan Citra dengan derau Coretean (c)

Pada citra yang mengalami derau titik dan derau garis seperti pada contoh Gambar 2.5a dan 2.5b, digunakan batasan atau *threshold* panjang dan lebar dari area yang telah disegmentasi.

Pada citra yang mengalami derau coretan atau *struckout* digunakan teknik *inpainting* atau menggambar ulang dengan menggunakan informasi ketetanggaan dari suatu pixel yang berisi karakter 'X'. Ketetanggan yang digunakan adalah 8-ketetanggan (*8-connected neighbors*). Penyelesaian ini telah digunakan oleh Axel dkk. sebagai salah satu fitur untuk memprediksi citra kata yang mengalami coretan [3]. Terdapat batasan (*treshold*) banyaknya ketetanggan yang digunakan, apabila kurang dari batasan tersebut maka pixel pada posisi tersebut yang awalnya memiliki karakter 'X' akan diubah menjadi karakter '.'. Penyelesaian ini berdasarkan pengamatan yang dilakukan bahwa garis coretan umumnya memiliki banyak ketetanggan yang sedikit dan tidak setebal garis yang dimiliki oleh digit.

2.3.1.2. Praproses Citra Digit yang Berimpitan

Pada permasalahan SPOJ *Hard Image Recognition* (HIR), terdapat beberapa citra yang memiliki beberapa digit yang berimpitan. Citra yang seperti ini akan bermasalah pada saat proses segmentasi karena akan terdeteksi hanya sebagai 1 digit saja.

Solusi dari permasalahan ini adalah dengan menggunakan banyaknya digit yang telah tersegmentasi. Karena keluaran dari permasalahan SPOJ *Hard Image Recognition* (HIR) pasti terdapat 6 digit, maka apabila banyak digit yang telah tersegmentasi kurang dari 6, maka dilakukan pemisahan digit pada daerah yang memiliki lebar lebih dari rata-rata lebar digit.

2.3.1.3. Normalisasi Ukuran Digit

Normalisasi ukuran atau proses *resize* adalah proses untuk merubah ukuran dari citra sesuai dengan ukuran yang diharapkan.

Normalisasi ukuran dilakukan pada digit-digit yang telah didapat agar digit-digit memiliki ukuran yang sama untuk menjadi masukan pada proses klasifikasi. Algoritma yang digunakan untuk melakukan normalisasi ukuran adalah *Nearest Neighbor Image Scaling*.

Nearest neighbor merupakan algoritma untuk melakukan perubahan ukuran dari citra yang sederhana dan cepat. *Nearest neighbor* tidak seperti algoritma lain, seperti *bilinear* dan *bicubic* yang menggunakan interpolasi untuk pixel tetangga sehingga akan menghasilkan citra yang lebih halus [4]. *Nearest neighbor* menggunakan interpolasi ketetanggaan untuk mendapatkan citra dengan ukuran baru, sehingga *Nearest neighbor* merupakan algoritma yang tepat untuk menyelesaikan permasalahan SPOJ *Hard Image Recognition* (HIR) yang merupakan citra biner, sehingga tidak ada nilai pixel baru yang dihasilkan pada proses interpolasi.

2.3.2. Klasifikasi Digit menggunakan JST

Jaringan Saraf Tiruan atau dalam bahasa Inggris disebut *Artificial Neural Network* merupakan pendekatan pengolahan informasi yang terinspirasi oleh cara kerja sistem saraf biologis, khususnya pada sel otak manusia dalam memproses informasi. Jaringan Saraf Tiruan terdiri dari sejumlah besar elemen pemrosesan informasi (*neuron*) yang saling terhubung dan bekerja sama untuk menyelesaikan sebuah masalah tertentu. Jaringan Saraf Tiruan akan melakukan proses pembelajaran untuk menghasilkan bobot terbaik.

Pendekatan yang akan digunakan untuk menyelesaikan permasalahan SPOJ *Hard Image Recognition* (HIR) ini adalah menggunakan bobot JST yang telah dilatih untuk menghitung luaran dari masukan citra pada permasalahan. Pendekatan dengan cara ini telah dilakukan oleh Cynthia pada permasalahan yang sama dengan melakukan percobaan beberapa kombinasi arsitektur secara manual dan dataset contoh citra masukan pada permasalahan SPOJ *Hard Image Recognition* (HIR) [5].

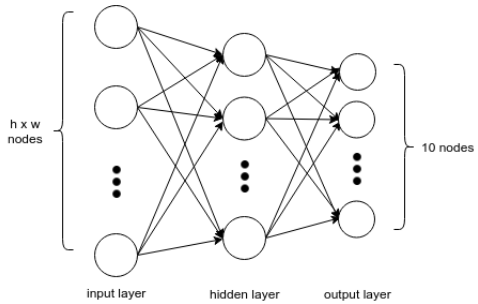
Terdapat dua bahasa pemrograman yang digunakan pada tugas akhir ini, yaitu bahasa pemrograman Python dan C++. Bahasa pemrograman Python digunakan untuk melatih JST menggunakan dataset yang dibuat. Bahasa pemrograman C++ digunakan sebagai pengujian dan diunggah ke SPOJ.

Dataset untuk pelatihan dibuat menggunakan font yang telah diunduh melalui website yang menyediakan berbagai jenis font [6]. Selain itu dataset juga diambil dari forum diskusi SPOJ *Hard Image Recognition* (HIR) [2]. Pada dataset terdapat digit dari 0 sampai dengan 9 menggunakan font yang berbeda dan dalam beberapa ukuran berbeda. Pembangunan dataset yang optimal dibutuhkan untuk meningkatkan akurasi.

Kemudian dilakukan proses pelatihan JST menggunakan dataset yang telah dibuat. Proses pelatihan JST dilakukan menggunakan program Python dan menggunakan *library* yang telah tersedia. Arsitektur JST yang digunakan hanya menggunakan 3 *layer* yang terdiri dari *layer input*, 1 *hidden layer*, dan *layer output*.

Layer input merupakan masukan yang berupa fitur dari citra. Fitur tersebut berupa nilai setiap pixel dari citra. Setiap citra masukan akan diubah ke dalam ukuran yang sama hxw. Proses perubahan ukuran ini menggunakan algoritma *Nearest Neighbor*. Kemudian layer input akan dikoneksikan pada *hidden* layer. *Hidden* layer memiliki n-*node* yang memiliki bobot dan bias berbeda. Nilai n pada *hidden* layer merupakan salah satu parameter yang dicari untuk mendapatkan luaran yang optimal. Kemudian *hidden* layer akan terhubung pada output layer. Output layer memiliki k-*node*, dimana k merupakan banyak kelas. Dalam permasalahan klasifikasi digit ini, terdapat kelas sebanyak 10, yaitu digit 0 sampai dengan 9. Setiap kelas akan memiliki nilai probabilitas yang apabila dijumlahkan sebesar 1. Kelas dengan nilai probabilitas terbesar akan menjadi kelas dari digit tersebut. Proses *backprogpagation* akan melakukan pelatihan untuk memperbaharui bobot dan bias pada setiap *node*. Setiap *node* juga memiliki fungsi aktivasi untuk membatasi besarnya

output. Selain itu setiap layer memiliki fungsi optimasi yang digunakan untuk memperbaharui bobot dan bias. Gambaran mengenai algoritma JST untuk klasifikasi digit terdapat pada Gambar 2.6.



Gambar 2.6 Struktur JST pada Klasifikasi Digit

Setelah mendapatkan bobot dan bias dari hasil pelatihan, bobot dan bias diekstrak untuk diletakkan pada program C++ sebagai pengujian. Pada program C++ terdapat proses awal (*preprocessing*) citra untuk mendapatkan digit dan proses perubahan ukuran sehingga semua digit memiliki ukuran yang sama sebelum masuk ke dalam JST. Proses perubahan ukuran juga menggunakan algoritma yang sama yang digunakan pada proses pelatihan, yaitu algoritma *Nearest Neighbor*. Pada program C++ ini terdapat proses menghitung luaran dari JST. Pada fungsi JST terdapat perkalian setiap node dengan bobot dan penambahan bias, kemudian terdapat fungsi aktivasi yang sama digunakan pada proses pelatihan, dan proses perhitungan luaran untuk mendapatkan kelas dengan probabilitas tertinggi. Perbedaan fungsi JST pada program pengujian C++ dengan program pelatihan Python adalah pada pada pengujian JST tidak terdapat proses pelatihan bobot karena bobot telah didapatkan dari program pelatihan. Program pengujian C++ ini disesuaikan dengan format input dan output dari permasalahan SPOJ *Hard Image Recognition* (HIR) kemudian diunggah ke dalam server SPOJ untuk mendapatkan nilai akurasi dari program yang telah dibangun.

2.3.2.1. Arsitektur Jaringan Saraf Tiruan

Pada jaringan saraf tiruan, setiap lapisan-lapisan (*layers*) memiliki sejumlah neuron. Arsitektur dari Jaringan Saraf Tiruan yang digunakan pada Tugas Akhir ini adalah arsitektur *Single-Layer Feedforward Networks*. Jaringan saraf ini hanya memiliki satu lapisan (*layer*) dengan bobot yang terhubung. Jaringan tersebut menerima input melalui lapisan tersembunyi. Jaringan ini hanya memiliki satu lapisan input dan satu lapisan output.

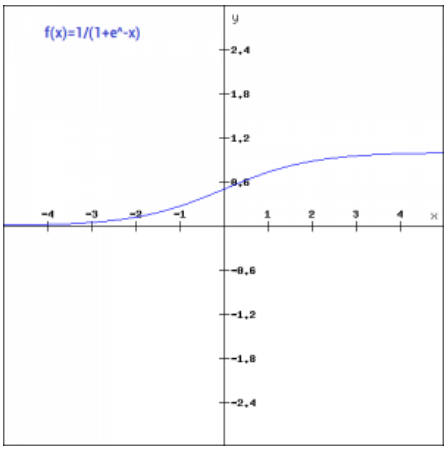
2.3.2.2. Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang berguna untuk membatasi amplitude atau besarnya output dari setiap neuron. Fungsi aktivasi juga disebut sebagai fungsi *squashing*, yaitu memperkecil rentang output signal dalam nilai yang terbatas [7]. Pada umumnya, rentang output yang telah dinormalisasi setelah dimasukkan ke dalam fungsi aktivasi akan berada dalam rentang $[0, 1]$ atau $[-1, 1]$. Pada tugas akhir ini fungsi aktivasi yang digunakan adalah fungsi aktivasi *logistic* atau yang disebut juga fungsi aktivasi *sigmoid*.

Fungsi aktivasi *sigmoid* merupakan fungsi aktivasi yang sering digunakan. Fungsi aktivasi ini dapat ditulis dalam persamaan 2.1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

Keuntungan dengan menggunakan fungsi aktivasi *sigmoid* adalah fungsi *sigmoid* merupakan fungsi yang non-linear. Dengan kata lain, apabila terdapat neuron-neuron yang memiliki fungsi *sigmoid* sebagai fungsi aktivasi, maka output dari neuron tersebut akan menjadi non-linear [8]. Fungsi aktivasi *sigmoid* memiliki rentang nilai 0-1 dan memiliki bentuk menyerupai huruf S, seperti pada Gambar 2.7.



Gambar 2.7 Grafik Fungsi Aktivasi *Sigmoid* [8]

2.3.2.3. Fungsi Optimasi

Fungsi optimasi merupakan fungsi yang digunakan untuk mengoptimalkan fungsi objektif. Pada jaringan saraf tiruan, fungsi optimasi digunakan untuk mengoptimalkan bobot dengan meminimalisasi *error* dari hasil prediksi terhadap output yang diinginkan. Pada tugas akhir ini, fungsi optimasi yang digunakan adalah fungsi optimasi *Adaptive Moment Estimation* (Adam).

Adam merupakan salah satu fungsi optimasi pada Jaringan Saraf Tiruan yang menggunakan *adaptive learning rate* untuk bobot dan bias dari setiap neuron. Berbeda dengan algoritma SGD yang menggunakan *learning rate* yang sama setiap saat [9], *learning rate* pada fungsi optimasi Adam beradaptasi saat pelatihan. Rumus gradien yang digunakan dapat dilihat pada persamaan 2.2. Kemudian gradien akan digunakan untuk memperbaharui bobot dan bias dengan menggunakan rumus pada persamaan 2.3 dan 2.4.

$$m_j = \beta_1 m_{j-1} + (1 - \beta_1) g_j \tag{2.2}$$

$$s_j = \beta_2 s_{j-1} + (1 - \beta_2)(g_j)^2 \tag{2.3}$$

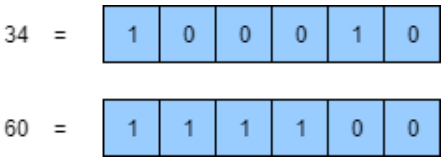
$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j} + \epsilon} m_j \tag{2.4}$$

2.3.3. Optimasi Arsitektur JST menggunakan GA

Penentuan nilai n pada *hidden layer* pada arsitektur JST yang telah dibangun merupakan parameter yang dapat dioptimasi. Tidak ada algoritma yang dapat menentukan berapa besar nilai n yang tepat pada masing-masing *hidden layer* pada JST. Besarnya n dapat bervariasi dari 1 hingga tak terbatas. Oleh karena itu digunakan algoritma optimasi yang tidak deterministik untuk menghitung besarnya nilai n yang tepat. Salah satu algoritma optimasi yang dapat digunakan adalah algoritma genetika atau *Genetic Algorithm*.

GA adalah teknik pencarian dalam bidang komputasi untuk menemukan solusi benar atau pendekatan untuk masalah optimasi dan pencarian. Teknik dalam GA didasarkan pada biologi evolusioner seperti pewarisan, mutasi, seleksi dan *crossover*.

Proses GA dimulai dengan menentukan populasi awal *initial population* yang terdiri dari beberapa kromosom yang disusun oleh beberapa gen yang merupakan representasi dari kandidat-kandidat solusi dari suatu masalah. Pada permasalahan SPOJ *Hard Image Recognition* (HIR) yang dikerjakan pada tugas akhir ini, sebuah kromosom berupa 6-*binary string* yang merupakan *encoding* dari besarnya n, yaitu banyak *neuron* dalam bentuk biner. Contoh pada Gambar 2.8 apabila kromosom berbentuk ”100010”, maka nilai dari n adalah 34, dan apabila kromosom berbentuk ”111100”, maka nilai dari n adalah 60.



Gambar 2.8 Contoh Kromosom pada Algoritma Genetika

2.3.3.1. Fitness Function

Fitness dalam GA didefinisikan sebagai gambaran kelayakan suatu solusi terhadap suatu permasalahan. *Fitness function* akan menghasilkan suatu nilai *fitness value* yang akan menjadi referensi untuk proses GA selanjutnya. Sebagai contoh pada permasalahan regresi, maka fungsi *fitness* yang cocok adalah dengan menghitung jarak antara nilai yang benar dengan nilai hasil prediksi. Pada permasalahan penentuan arsitektur yang baik untuk Jaringan Saraf Tiruan, Benardos dan Vosniakos menggunakan fungsi *fitness* dengan menghitung *error* dari *training* ditambah dengan *error* dari *generalization* atau *validation* [10].

Pada permasalahan SPOJ *Hard Image Recognition* (HIR), penulis menggunakan akurasi dari data *training* ditambah dengan data *testing* atau yang disebut dengan *generalization*. Untuk menambah rentang nilai, penulis menggunakan pemangkatan dengan bilangan e seperti pada persamaan 2.5.

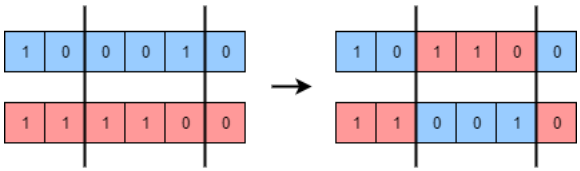
$$F(i) = e^{A_{training}(i)+A_{generalization}(i)} \tag{2.5}$$

2.3.3.2. Crossover

Crossover adalah suatu operator rekombinasi yang memiliki tujuan untuk mendapatkan individu baru yang lebih baik. Operator *crossover* akan melakukan rekombinasi dari kumpulan *parents* yang telah terbentuk dari proses seleksi. Proses *crossover* akan dilakukan sehingga menghasilkan banyak populasi yang sama dikurangi

dengan banyak *parents* yang terpilih. *Crossover* akan diaplikasikan pada dua kromosom yang terpilih dengan proses sebagai berikut [11]:

1. Pilih titik pada kromosom secara acak.
2. Bagi kromosom menjadi dua bagian, pemisahan dilakukan pada titik yang telah terpilih.
3. Satukan kembali kromosom tersebut dengan bagian depan kromosom pertama akan menjadi bagian depan pada kromosom kedua dan bagian depan dari kromosom kedua akan menjadi bagian depan pada kromosom pertama, seperti penukaran.

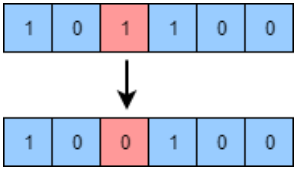


Gambar 2.9 Proses *Crossover* pada *Genetic Algorithm*

Pada Gambar 2.9 proses *crossover* dilakukan pada kedua *parents* dengan nilai n sebesar 34 dan 60, yang kemudian menghasilkan 2 individu baru dengan nilai n sebesar 44 dan 50.

2.3.3.3. Mutation

Mutasi merupakan proses pada GA yang bertujuan untuk mempertahankan keragaman dengan menukar salah satu atau lebih gen dalam kromosom dengan nilai kebalikannya. Probabilitas dilakukan mutasi pada gen sangat kecil, biasanya hanya sekitar 0.01 atau 0.001 [11]. Pada permasalahan ini, mutasi akan dilakukan pada 1 gen pada setiap kromosom hasil proses *crossover*.



Gambar 2.10 Proses *Mutation* pada *Genetic Algorithm*

Pada Gambar 2.10, proses mutasi dilakukan pada salah satu gen pada kromosom dengan nilai n sebesar 44 dan menghasilkan individu baru dengan nilai n sebesar 36.

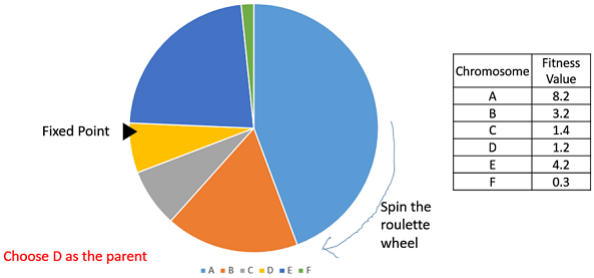
2.3.3.4. Parent Selection

Proses pemilihan *parents* merupakan proses memilih individu-individu terbaik yang akan menjadi *parents* untuk mendapatkan individu-individu baru yang akan digunakan pada generasi berikutnya. Harapannya individu-individu baru tersebut akan memiliki sifat yang sama dengan sifat yang dimiliki oleh orang-tuanya. Salah satu algoritma yang umum digunakan dalam pemilihan *parents* adalah *Roulette Wheel Selection* (RWS).

Algoritma RWS diumpamakan seperti memutar roda putar dimana *parent* dipilih di titik yang ditunjuk pada saat roda tersebut berhenti. Kemudian pemutaran dilakukan kembali untuk *parent* kedua dan seterusnya. Pada algoritma RWS setiap kromosom akan memiliki peluang yang berbeda untuk terpilih, sesuai dengan besar nilai *fitness* yang dimiliki kromosom tersebut, sehingga apabila kromosom tersebut memiliki nilai *fitness* yang besar, maka semakin besar peluang kromosom tersebut terpilih sebagai *parent*. Ilustrasi dari algoritma RWS pada 2.11.

Pseudocode algoritma *Roulette Wheel Selection* dapat dilihat pada *pseudocode* 2.1.

Akan tetapi, algoritma RWS hanya dapat memilih satu titik, sementara untuk melakukan proses *crossover* pada regenerasi ke-



Gambar 2.11 Algoritma *Roulette Wheel Selection* [12]

Pseudocode 2.1: Algoritma ROTATE WHEEL SELECTION

```
1: function RWS(POPULATION)
2:     F ← total fitness in Population
3:     R ← random number between [0..F]
4:     i = 0
5:     while fitness sum of Population[0..i] < R do
6:         i ← i + 1
7:     end while
8:     return Population[i]
9: end function
```

turunan, diperlukan setidaknya dua titik (*parents*). Salah satu algoritma yang dapat digunakan yaitu algoritma *Stochastic Universal Sampling* (SUS). Algoritma SUS merupakan pengembangan dari algoritma RWS. Pada algoritma SUS titik pada roda putar dapat dipilih lebih dari satu titik sehingga dapat memilih lebih dari satu *parent* dengan *parent* yang sama tidak dapat dipilih lebih dari satu kali. *Pseudocode* algoritma *Stochastic Universal Sampling* dapat dilihat pada *pseudocode* 2.2 [13].

Proses akan dilakukan untuk mendapatkan n optimal dengan menjalankan beberapa generasi. Pada setiap generasi akan dihitung nilai *fitness* yang akan digunakan untuk melakukan pemilihan *pa-*

Pseudocode 2.2: Algoritma STOCHASTIC UNIVERSAL SAMPLING

```
1:  function SUS(POPULATION, N)
2:      F ← total fitness of Population
3:      N ← number of offspring to keep
4:      P ← distance between the pointers (F/N)
5:      Start ← random number between 0 and P
6:      Pointers ← [Start + i * P|i in [0..(N − 1)]]
7:      return RWS(POPULATION, POINTERS)
8:  end function
9:  function RWS(POPULATION, POINTS)
10:     Keep = []
11:     for P in Points do
12:         i ← 0
13:         while fitness sum of Population[0..i] < P do
14:             i ← i + 1
15:             add Population[i] to Keep
16:         end while
17:     end for
18:     return Keep
19: end function
```

rents. Kemudian dari *parents* yang terpilih akan dilakukan proses *crossover* dan mutasi. Proses ini akan dilakukan berulang-ulang sampai banyak maksimum generasi terpenuhi atau nilai *fitness* tidak mengalami perubahan. Hasil n optimal yang didapat akan digunakan para proses pelatihan JST.

2.4. Deskripsi Umum

Pada subbab ini akan dijelaskan definisi, deskripsi dan landasan yang akan digunakan pada penyelesaian masalah.

2.4.1. Online Judge

Online Judge atau sistem penilaian daring merupakan sistem untuk mengetes program secara otomatis dalam soal pemrograman. Sistem ini berisi soal-soal pemrograman dimana sistem akan menilai apakah sebuah solusi tertentu dapat menyelesaikan soal-soal yang berada dalam sistem tersebut. Sistem dapat mengkompilasi, mengeksekusi, dan membandingkan keluaran dari program atau solusi yang diunggah dengan keluaran yang seharusnya. Sistem akan mengembalikan hasil kepada pengguna apakah kode program yang diunggah diterima atau tidak. Terdapat banyak kategori *online judge* yang saat ini telah diimplementasikan, salah satunya *problem solving online judge* yang digunakan oleh para programmer untuk mengasah kemampuan logikanya dalam menyelesaikan *problem set*, salah satu contohnya adalah *Sphere Online Judge* atau SPOJ yang digunakan untuk melakukan uji coba pada tugas akhir ini.

2.4.2. Pengenalan Pola Citra

Pengenalan pola atau dalam Bahasa Inggris adalah *pattern recognition* mencakup berbagai permasalahan pemrosesan informasi seperti pengenalan huruf pada tulisan tangan dan pengenalan wajah manusia. Kegiatan pengenalan pola adalah proses pengelompokan data numerik maupun simbolik (termasuk citra) yang dilakukan secara otomatis menggunakan mesin komputer. Tujuan dari pengenalan pola ini adalah untuk mengenali suatu objek dalam citra. Manusia dapat dengan mudah mengatasi permasalahan tersebut tapi tidak dengan komputer. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Hal inilah yang akan diterapkan pada sistem komputer.

Komputer akan menerima masukan yang berupa citra dari objek yang akan diidentifikasi. Kemudian komputer akan memproses

masukan citra tersebut dan memberi keluaran yang berupa informasi yang terdapat pada citra tersebut.

Gambar 2.12 Proses Pengenalan Pola pada Citra

Salah satu contoh permasalahan pada pengenalan pola adalah klasifikasi digit. Citra dari sebuah digit dimasukkan dan komputer mencari algoritma yang dapat membedakan digit tersebut. Terdapat 10 kelas yang mewakili digit 0 sampai 9. Agar komputer dapat mengenali pola tersebut, dibutuhkan *dataset* yang disiapkan oleh manusia yang berupa citra untuk digit 0 sampai dengan 9 dengan label atau kelas yang sudah ditentukan sebelumnya.

2.4.3. Tipografi

Tipografi atau tata huruf merupakan suatu teknik memilih dan menata huruf, kata, paragraf pada ruang-ruang yang tersedia untuk menciptakan kesan tertentu agar pembaca lebih nyaman. Dalam beberapa lieteratur tipofrafi, rupa huruf dapat digolongkan dalam beberapa klasifikasi, yang berguna untuk mempermudah mengidentifikasi rupa huruf tersebut [14]. Berdasarkan klasifikasi yang umum dan sering dipakai, klasifikasi yang berdasarkan bentuk rupa hurufnya, rupa huruf dibagi menjadi:

1. Roman, merupakan seluruh huruf yang mempunyai ciri tegak dan didominasi garis lurus kaku. Roman dibagi lagi menjadi Serif, Egyptian dan Sans Serif. Serif dengan ciri memiliki siripan di ujungnya. Egyptian, atau populer dengan sebutan slab serif memiliki ciri kaki/sirip yang berbentuk persegi seperti papan dengan ketebalan yang sama atau hampir sama. Sans serif mimiliki ciri tanpa sirip/serif dan memiliki ketebalan huruf yang sama atau hampir sama dan kesan yang

ditimbulkan oleh huruf ini adalah modern, kontemporer dan efisien.

2. Script, merupakan goresan tangan yang dikerjakan dengan pena, kuas atau pensil tajam dan biasanya miring ke kanan. Kesan yang ditimbulkan adalah sifat pribadi dan akrab.
3. Miscellaneous, merupakan pengembangan dari bentuk-bentuk yang sudah ada. Ditambah dengan hiasan dan ornamen atau garis-garis dekoratif. Kesan yang dimiliki adalah dekoratif dan ornamental.

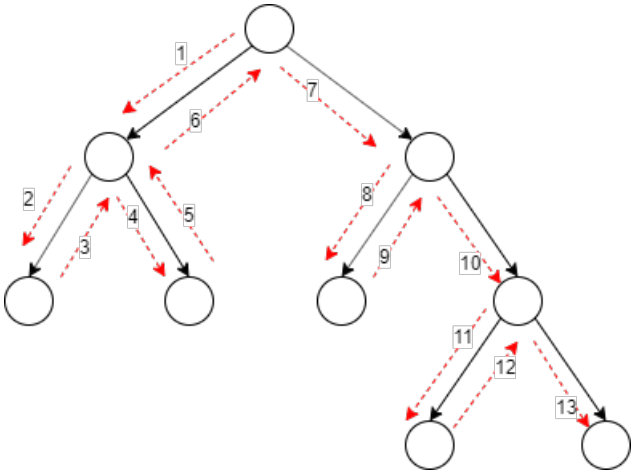
2.4.4. Segmentasi Citra

Segmentasi merupakan proses untuk mempartisi citra menjadi beberapa daerah atau objek dengan setiap objek memiliki kemiripan antara pixel dengan pixel tetangganya. Pada permasalahan SPOJ *Hard Image Recognition* (HIR), masukan yang berupa citra digit akan dilakukan proses segmentasi untuk memisahkan karakter dengan latar belakangnya dan untuk mendapatkan posisi dari karakter-karakter yang diduga sebagai digit yang akan digunakan pada tahap proses pengenalan karakter digit. Contoh hasil dari proses segmentasi karakter terdapat pada contoh Gambar 2.13.

Gambar 2.13 Contoh hasil segmentasi pada citra digit

2.4.5. Depth First Search (DFS)

Depth First Search (DFS) merupakan salah satu teknik pencarian secara mendalam pada sebuah graf. DFS mengeksplor setiap edge yang berhubungan dengan vertex v yang terakhir dikunjungi. Setelah setiap edge dari vertex v telah dikunjungi, pencarian akan melakukan *backtracking* untuk mengeksplorasi edge dari vertex v sebelumnya yang masih belum dikunjungi. Proses ini akan dilakukan sampai semua vertex telah dikunjungi dari vertex awal (*source*). Apabila masih ada vertex yang masih belum dikunjungi, algoritma dfs akan memilih salah satu dari vertex tersebut untuk menjadi *source* baru. Proses akan dilakukan sampai semua vertex pada graf telah dikunjungi [15]. Ilustrasi *depth first search* terdapat pada Gambar 2.14.



Gambar 2.14 Ilustrasi Algoritma DFS

2.4.6. Penghilangan Noise

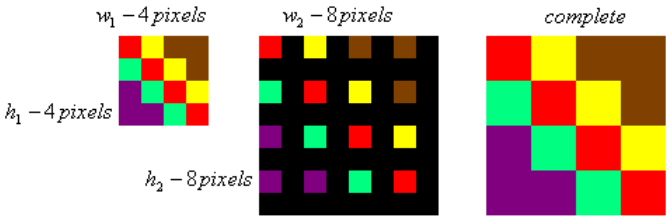
Noise atau derau adalah gangguan yang terdapat pada data digital salah satunya yang berupa citra, yang disebabkan oleh faktor

internal atau faktor eksternal sehingga dapat mengurangi kualitas dari data tersebut. Derau pada gambar dapat disebabkan oleh beberapa hal seperti adanya gangguan kotoran pada kamera, adanya kesalahan pada proses pengolahan dan dapat pula disebabkan karena adanya coretan tangan atau tumpahan tinta. Pada permasalahan SPOJ *Hard Image Recognition* (HIR), terdapat citra yang memiliki *noise* yang berupa coretan, berupa bintik-bintik kecil dan berupa *border* atau garis tepi. Pada tugas akhir ini *noise* yang terdapat pada citra akan dihilangkan sehingga dapat meningkatkan akurasi pada pengenalan digit.

2.4.7. Nearest Neighbor Image Scalling

Nearest neighbor merupakan algoritma sederhana dan cepat untuk melakukan proses merubah ukuran dari citra. *Nearest neighbor* tidak seperti algoritma lain, seperti *bilinear*, *bicubic*, *spline* dan lainnya, yang menggunakan interpolasi untuk pixel tetangga yang menghasilkan citra yang lebih halus atau lebih baik [4]. Prinsip dari perubahan ukuran citra adalah untuk mendapatkan citra baru dengan ukuran baru dengan menggunakan citra lama sebagai dasar. Citra baru dapat lebih kecil, lebih besar atau sama besarnya bergantung pada perbandingan ukuran. Saat memperbesar suatu gambar, yang kita lakukan adalah menambahkan kotak kosong pada citra dasar. Contoh pada Gambar 2.15 untuk pembesaran citra menggunakan interpolasi ketetanggaan.

Algoritma perubahan ukuran citra bertujuan untuk meletakkan kotak kosong dan kemudian mengisi kotak kosong tersebut dengan interpolasi warna yang tepat. Pada teknik *nearest neighbor*, kotak kosong tersebut akan diisi dengan *pixel* tetangga. Algoritma *nearest neighbor* dapat melakukan pembesaran maupun pengecilan citra. Untuk melakukan perubahan ukuran citra diperlukan *horizontal ratio* dan *vertical ratio* pada citra. *Horizontal ratio* didapatkan dari perubahan atau perbandingan antara tinggi awal dengan tinggi akhir, sementara *vertical ratio* didapatkan dari perbandingan antara



Gambar 2.15 Perubahan ukuran citra

lebar awal dengan lebar akhir.

$$\left.\begin{aligned} xratio &= \frac{w_1}{w_2} \\ yratio &= \frac{h_1}{h_2} \end{aligned}\right\} w_2, h_2 \neq 0$$

Pada *pseudocode* 2.3 merupakan fungsi dari algoritma *nearest neighbor*.

Pseudocode 2.3: Algoritma NEAREST NEIGHBOR

```
1:  INTEGER w1, h1, w2, h2
2:  ARRAY pixels[w1 * h1], temp[w2 * h2]
3:  DOUBLE xratio ← w1/ (double) w2
4:  DOUBLE yratio ← h1/ (double) h2
5:  DOUBLE px, py
6:  for i ← 0, h2 do
7:      for j ← 0, w2 do
8:          px ← ⌊j * xratio⌋
9:          py ← ⌊i * yratio⌋
10:         temp[(i * w2) + j] ← pixels[(int)((py * w1) + px)]
11:     end for
12: end for
```

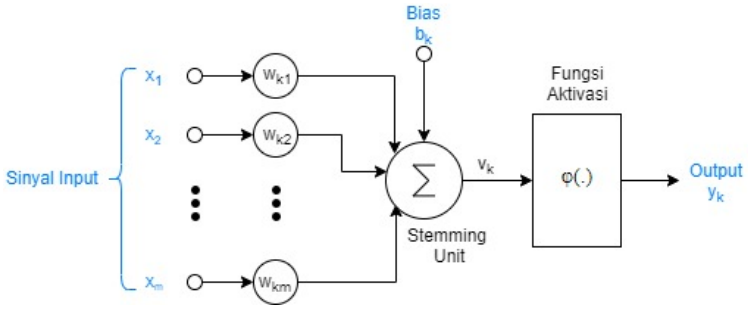
2.4.8. Jaringan Saraf Tiruan

Jaringan Saraf Tiruan atau dalam bahasa inggris disebut *Artificial Neural Network* adalah sebuah sistem pararel yang terdistribusi secara masif yang terdiri dari unit pemrosesan yang sederhana yang memiliki kecendurungan alami untuk menyimpan pengetahuan berdasarkan pengalaman dan menggunakan pengetahuan tersebut [7]. Jaringan saraf tiruan merupakan representasi buatan dari otak manusia yang selalu melakukan proses pembelajaran. Dalam jaringan saraf tiruan terdapat tiga elemen dasar dan berperan penting [7], yaitu sebagai berikut:

1. Synapsis (Jalur Penghubung) antara neuron yang masing-masing memiliki bobot. Setiap sinyal x_j pada input sinapsis j yang terkoneksi dengan neuron k akan dikalikan dengan bobot sinapsis w_{kj}.
2. *Adder* atau *summing unit* untuk menjumlahkan total signal input.
3. Fungsi aktivasi untuk membatasi besar output dari neuron yang bervariasi.

Secara umum, terdapat beberapa lapisan pada jaringan saraf tiruan, yaitu lapisan masukan dan lapisan keluaran. Diantara lapisan masukan dan lapisan luaran terdapat lapisan tersembunyi. Lapisan masukan merupakan fitur-fitur yang merupakan representasi dari sebuah data, contohnya pada sebuah citra, fitur dapat berupa nilai piksel-piksel pada citra ataupun fitur lainnya seperti luas, keliling, dan sebagainya. Lapisan tersembunyi yang terletak di antara lapisan masukan dan lapisan luaran akan menerima nilai yang sudah diproses dengan bobot pada masukan dan menghasilkan luaran. Lapisan luaran adalah lapisan terakhir pada jaringan yang menghasilkan luaran dari program, luaran dapat berupa kelas atau kategori dari data apabila dalam permasalahan klasifikasi. Model dari neuron pada jaringan saraf tiruan digambarkan pada Gambar 2.16.

Dalam notasi matematika, perhitungan luaran pada neuron k



Gambar 2.16 Ilustrasi Jaringan Saraf Tiruan

dilakukan dengan persamaan 2.6, kemudian setelah ditambahkan bias dan fungsi aktivasi seperti pada persamaan 2.7.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.6}$$

$$y_k = \varphi(u_k + b_k) \tag{2.7}$$

Dimana x_1, x_2, ..., x_m merupakan sinyal input; w_{k1}, w_{k2}, ..., w_{km} merupakan bobot dari neuron k; u_k merupakan penjumlahan fungsi linear output dari sinyal input; b_k merupakan bias; φ merupakan fungsi aktivasi; dan y_k merupakan sinyal output dari neuron. Tujuan dari penambahan bias b_k adalah untuk mengaplikasikan transformasi *affine* pada output.

Jaringan saraf tiruan melakukan pelatihan dengan cara:

1. Menghitung output
2. Membandingkan output yang diterima dengan target yang diinginkan
3. Menyesuaikan bobot dan mengulangi proses.

2.4.9. Genetic Algorithm

Genetic Algorithm (GA) adalah teknik pencarian dalam bidang komputasi untuk menemukan solusi benar atau pendekatan untuk masalah optimasi dan pencarian. GA merupakan bagian dari *Evolutionary Algorithm*, yaitu suatu algoritma yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah [16]. Teknik dalam GA didasarkan pada teknik pada biologi evolusioner seperti pewarisan, mutasi, seleksi, dan *crossover*.

Proses GA dimulai dengan menentukan populasi awal yang terdiri dari beberapa kromosom yang disusun oleh beberapa gen yang merupakan representasi dari kandidat-kandidat solusi dari suatu masalah. Kandidat-kandidat terbaik akan dipilih melalui proses seleksi, berdasarkan *fitness value* yang telah dihitung untuk setiap kromosom dalam populasi. Kandidat–kandidat terpilih dari proses ini adalah individu -individu yang akan mengisi *mating pool* yaitu suatu set dimana dua *parents* akan dibentuk dari sini. Dalam *Evolutionary Algorithm* prinsip bertahan muncul karena adanya proses reproduksi. Turunan *offspring* yang dihasilkan akan membawa sifat gen orang tuanya (*parents*), oleh sebab itu *parents* dipilih dari *mating pool* yang merupakan kumpulan kandidat-kandidat terbaik dari suatu populasi. Dengan demikian turunan yang dihasilkan adalah turunan yang memiliki sifat unggul dari kedua orang tuanya.

Proses dari *Genetic Algorithm* adalah sebagai berikut [11]:

1. Generasi populasi awal yang terdiri dari kromosom-kromosom secara random.
2. Hentikan apabila kriteria untuk proses diberhentikan telah terpenuhi, jika tidak lanjut ke proses 3.
3. Hitung *fitness value* dari setiap kromosom.
4. *Crossover* dan *mutation* diaplikasikan pada kromosom yang terpilih pada generasi saat itu untuk membuat populasi baru

yang akan digunakan pada generasi selanjutnya.

5. Kembali ke proses 2.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN

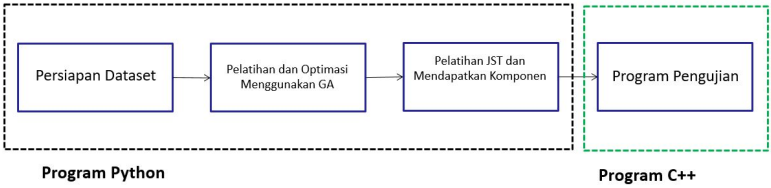
Pada bab ini akan dijelaskan desain algoritma yang akan digunakan untuk menyelesaikan permasalahan.

3.1. Deskripsi Umum Sistem

Permasalahan pada SPOJ *Hard Image recognition* (HIR) akan diselesaikan menggunakan algoritma Jaringan Saraf Tiruan dengan optimasi algoritma Genetika atau. Pada permasalahan optimasi arsitektur Jaringan Saraf Tiruan, Benardos dan Vosniakos menggunakan GA untuk mengoptimasi arsitektur JST dengan mencari banyak *hidden layer* dan banyak *neuron* pada tiap *hidden layer* [10].

Penyelesaian pada permasalahan tugas akhir ini akan dilakukan dengan menggunakan dua bahasa pemrograman, yaitu dengan bahasa pemrograman Python dan C++. Program dengan bahasa pemrograman Python digunakan untuk membangun fungsi pembuatan dataset, pelatihan Jaringan Saraf Tiruan serta optimasi dengan menggunakan algoritma Genetika. Program dengan bahasa pemrograman C++ digunakan untuk pengujian JST dengan citra-citra masukan dari SPOJ.

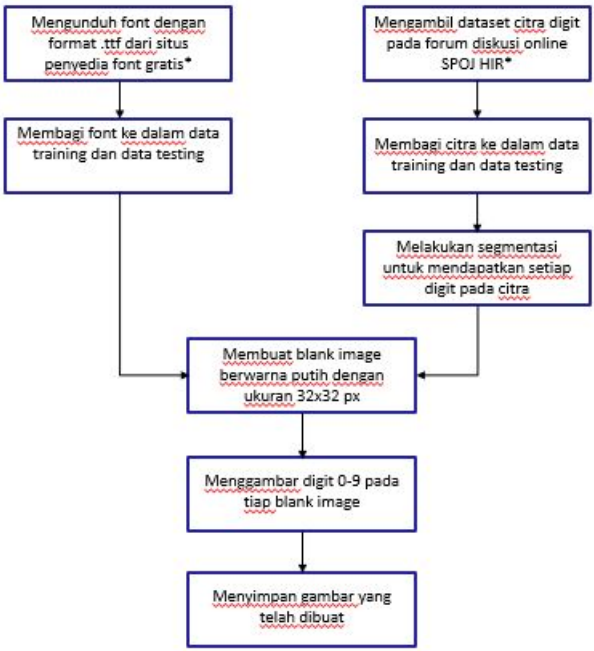
Gambar 3.1 merupakan proses penyelesaian secara umum permasalahan SPOJ *Hard Image Recognition* (HIR).



Gambar 3.1 Diagram Umum Penyelesaian

3.2. Desain Pembuatan Dataset

Langkah awal yang dilakukan sebelum melakukan proses pelatihan adalah mempersiapkan citra pelatihan. Dataset citra didapatkan dengan membangun fungsi yang menggambar digit 0 sampai 9 dengan menggunakan 60 jenis *font* yang berbeda yang didapat dari [6]. Data pelatihan juga dilakukan dengan melakukan segmentasi secara manual dengan menggunakan dataset yang diambil dari [2]. Gambar 3.2 merupakan diagram proses pembuatan dataset.



Gambar 3.2 Proses Pembuatan Dataset

Sebanyak 60 jenis *font* berbeda diunduh dari [6]. Ke-60 jenis *font* tersebut dibagi dalam proporsi 5:1 untuk data latih (data *training*) dan data uji (data *testing*). Gambar 3.3 merupakan contoh dari

font yang telah diunduh. Detail mengenai nama dan jenis *font* yang digunakan terdapat pada lampiran.

Gambar 3.3 Contoh *Font* yang Telah Diunduh

Proses mempersiapkan dataset ini dilakukan menggunakan bahasa pemrograman Python dengan menggunakan library Pillow. Setelah mempersiapkan jenis *font* yang akan digunakan, sebuah *blank image* berwarna putih sebesar 32x32 pixel dibuat untuk menjadi *background* dari citra digit. Citra digit dari 0-9 digambar pada *background* tersebut dengan 3 ukuran yang berbeda. *Pseudocode* untuk pembuatan data *train* dan data *test* terdapat pada *pseudocode* 3.1. Daftar nama variabel beserta fungsinya terdapat pada Tabel 3.1.

Tabel 3.1 Nama dan Fungsi Variabel dalam Pembuatan Data *Training* dan *Testing*

Nama Variabel	Fungsi Variabel
$path$	Menyimpan semua data font yang memiliki format **.ttf** pada suatu folder
$fonts$	*Dictionary* dengan *key* berupa kode font dan *value* berupa path dari font
$size$	*Array* ukuran dari tipografi yang diinginkan
$file\_out$	Nama file yang digunakan untuk menyimpan citra digit yang telah digambar pada setiap iterasi. Citra akan disimpan ke dalam folder *training* dan *testing*

Selain membuat dataset menggunakan font yang telah diunduh, penulis juga menggunakan dataset yang terdapat pada [2]. Penulis melakukan segmentasi secara manual pada bagian-bagian yang merupakan digit pada masing-masing citra. Untuk setiap citra, citra akan dilakukan proses *threshold* dengan tujuan agar citra hanya memiliki dua buah nilai, yaitu berwarna hitam atau putih. Kemudian posisi-posisi titik kiri-atas, titik kanan-atas, titik kiri-bawah, dan titik kanan-bawah disimpan secara manual untuk masing-masing digit, sehingga total terdapat 6 posisi yang berbeda. *Pseudocode* untuk proses segmentasi secara manual pada citra HIR terdapat pada *pseudocode* 3.2. Daftar nama variabel beserta fungsinya terdapat pada Tabel 3.2

Pseudocode 3.1: Pembuatan Data Training

```
path ← get all font path for data training
fonts ← dictionary {}
size ← [24, 20, 16]
for f in path do
    fonts[f.last_name] ← f
end for
for i in range (0..10) do
    for f in fonts do
        for s in size do
            file_out ← "dataset/train/i_f_s.png"
            DRAW_DIGIT(file_out, i, fonts[f], s)
        end for
    end for
end for
```

Pseudocode 3.2: Penambahan Dataset HIR

```
path ← get all HIR image path
img ← image_read(path, grayscale)
ret, img_th ← cv2.threshold(img, binary_inv)
dgt_list ← Array [] of 6 digit position
i ← 0
for d in dgt_list do
    filename ← "dataset/test/hir_001_i.png"
    HIR_DIGIT(img_th, d, filename)
    i ← i + 1
end for
```

3.2.1. Desain Fungsi Menggambar Digit

Fungsi ini merupakan fungsi untuk menggambar digit dari 0 sampai dengan 9 dengan font yang telah diberikan. Pada fungsi ini

Tabel 3.2 Nama dan Fungsi Variabel dalam Pembuatan Data Segmentasi Citra HIR

Nama Variabel	Fungsi Variabel
$path$	Menyimpan semua data citra HIR yang memiliki format **.jpg** pada suatu folder
img	Menyimpan nilai setiap pixel dari suatu citra dalam bentuk array
img_th	Menyimpan nilai citra setelah dilakukan *thresholding*
dgt_list	Array untuk menyimpan posisi dari setiap digit
$filename$	Menyimpan nama file yang digunakan untuk menyimpan hasil segmentasi secara manual untuk setiap digit

menggunakan *library* tambahan yaitu *PIL* atau *Pillow*, yang membantu proses menggambar. Fungsi ini digambarkan pada *pseudocode* 3.3. Masukan, proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.3.

Pseudocode 3.3: Menggambar Digit menggunakan Font

```
1: function DRAW_DIGIT(file_out, digit, font_type, size)
2:     img ← 32x32 white blank image
3:     font ← load_font(font_type)
4:     w, h ← font.getsize()
5:     draw ← DRAW(img)
6:     draw.text((32 − w/2), (32 − h/2), digit, font, black)
7:     img.save(file_out)
8: end function
```

Tabel 3.3 Masukan, Proses, dan Keluaran dari Fungsi Menggambar Digit Menggunakan Font

Masukan	Proses	Keluaran
String $file\_out$ yang menyatakan *output path*, bilangan bulat $digit$ yang menyatakan digit yang akan ditulis, *String* $font\_type$ yang menyatakan *font path*, dan bilangan bulat $size$ yang menyatakan besar karakter digit	Membuat citra digit sesuai dengan font dan ukuran yang ditentukan	Citra digit dalam format .png

3.2.2. Desain Fungsi Mengsegmentasi Digit

Fungsi ini merupakan fungsi untuk melakukan segmentasi pada masing-masing digit pada citra HIR sesuai dengan posisi digit tersebut. Pada fungsi ini terdapat proses untuk mendapatkan area terbesar dari suatu daerah, sehingga area terbesar tersebut dianggap sebagai digit dan *noise* atau bagian yang bukan merupakan digit dapat dihilangkan. Pada fungsi ini menggunakan *library* tambahan yaitu *OpenCV 2* untuk membantu proses segmentasi. Hasil dari segmentasi akan disalin ke *blank image* dan kemudian disimpan. Fungsi ini digambarkan pada *pseudocode* 3.4. Masukan, proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.4.

Tabel 3.4 Masukan, Proses, dan Keluaran dari Fungsi HIR DIGIT

Masukan	Proses	Keluaran
String img yang menyatakan citra HIR yang telah dibaca sistem, *array* yang berisi 4 bilangan bulat pos yang menyatakan posisi digit, *String* $filename$ yang menyatakan *output path*	Menyalin citra digit pada citra HIR sesuai ukuran yang diharapkan	Citra digit dalam format .png

Pseudocode 3.4: Segmentasi Digit HIR

```
1:  function HIR_DIGIT(img, pos, filename)
2:      new_dgt ← 32x32 white blank image
3:      dgt ← img[pos]
4:      ret, dgt ← threshold(127, 255)
5:      dgt_resized ← resize(20, 30, inter_nearest)
6:      new_dgt ← dgt_resized
7:      ret, labels ← connectedComponents(temp)
8:      labels_props ← measure.regionprops(labels)
9:      max_area, dgt_index ← 0
10:     for label_props in labels_props do
11:         if label_props.area > max_area then
12:             max_area ← label_props.area
13:             dgt_index ← label_props.index
14:         end if
15:     end for
16:     dgt_mask ← labels == dgt_index + 1
17:     new_dgt[dgt_mask == False] ← 255
18:     new_dgt.save(filename)
19: end function
```

3.3. Desain Pelatihan dan Optimasi menggunakan Genetic Algorithm

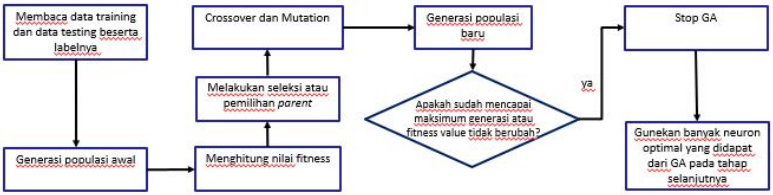
Setelah mempersiapkan dataset, dilakukan proses pelatihan menggunakan algoritma jaringan saraf tiruan yang akan dioptimasi menggunakan algoritma genetika. Pada tugas akhir ini, algoritma genetika digunakan untuk mencari parameter banyak *neuron* terbaik pada *hidden layer*. Banyak *hidden layer* yang pada tugas akhir ini terbatas sebanyak 1 buah *hidden layer*.

Proses GA dimulai dengan menentukan populasi awal yang terdiri dari beberapa kromosom yang disusun oleh beberapa gen yang merupakan representasi dari solusi dari permasalahan. Pada permasalahan tugas akhir ini, banyak populasi yang digunakan adalah 8 *chromosome*, dengan masing-masing *chromosome* berbentuk 6-*binary string* yang akan menyatakan banyak neuron pada *hidden layer*. Contoh apabila *chromosome* tersebut berbentuk ”100010”, maka banyak neuron adalah 34.

Fungsi *fitness* yang digunakan adalah nilai akurasi pada data *training* ditambah nilai akurasi pada data *testing*. Sehingga, semakin tinggi nilai akurasi, maka nilai *fitness* yang dihasilkan semakin tinggi dan semakin besar kemungkinan *chromosome* tersebut menjadi kandidat terbaik. Masing-masing kandidat akan dihitung nilai *fitness* nya kemudian 2 kandidat terbaik akan dipilih sebagai *parent* menggunakan algoritma *Stochastic Universal Sampling*.

Pada proses evolusi, dilakukan crossover dan *mutation*. Proses *crossover* dilakukan pada sebanyak 6 *chromosome*. Proses mutasi dilakukan pada 1 buah gen pada *chromosome*. Proses ini dilakukan terus menerus sampai maksimal 300 generasi atau berhenti pada saat nilai *fitness* tidak berubah selama 5 kali generasi.

Gambar 3.4 merupakan diagram proses pelatihan dan optimasi menggunakan *Genetic Algorithm*.



Gambar 3.4 Proses Optimasi menggunakan Genetic Algorithm

Pseudocode 3.5 merupakan program utama untuk menjalankan *Genetic Algorithm*. Masukan, proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.5.

Tabel 3.5 Masukan, Proses, dan Keluaran dari Fungsi Genetic Algorithm

Masukan	Proses	Keluaran
String $train\_x, test\_x$ yang menyatakan citra dataset training dan testing yang telah dibaca sistem, $train\_y, test\_y$ yang menyatakan label dari dataset training dan testing (0 - 9)	Proses evolusi *Genetic Algorithm*	Hasil neuron terbaik pada populasi pada tiap generasi

Pseudocode 3.5: Fungsi GENETIC ALGORITHM

```
1:  function GENETIC_ALGORITHM(train_x, train_y, test_x, test_y)
2:      pop_size, n_gene, n_generation, n_parents
3:      pop ← generate pop_size random chromosome
4:      for generation in range n_generation do
5:          obj_val ← calc_objective_val( pop)
6:          idx ← index_max(obj_val)
7:          hidden_layer ← decode(pop[idx])
8:          parents ← select_parents(pop, obj_val, n_parents)
9:          off_cross ← crossover(not parents, n_cross)
10:         off_mut ← mutation( off_cross, n_flip )
11:         pop[: n_parents] ← parents
12:         pop[n_parents :] ← off_mut
13:     end for
14: end function
```

3.3.1. Desain Fungsi Menghitung Nilai Objektif

Fungsi ini merupakan fungsi untuk menghitung nilai *fitness* atau nilai objektif dari optimasi *genetic algorithm* pada *neural network*. Fungsi objektif yang digunakan adalah memaksimalkan nilai akurasi dari data *training* dan data *testing*. Sehingga nilai *fitness* didapat dari bilangan e yang dipangkatkan dengan penjumlahan akurasi pada data *training* ditambah dengan akurasi pada data *testing* atau disebut dengan *generalization accuracy* seperti pada persamaan 3.1.

$$F(i) = e^{A_{training}(i)+A_{generalization}(i)} \tag{3.1}$$

Desain fungsi menghitung nilai *fitness* dari individu-individu pada populasi terdapat pada *pseudocode* 3.6. Masukan, proses, dan keluaran pada fungsi terdapat pada Tabel 3.6.

Tabel 3.6 Masukan, Proses, dan Keluaran dari Fungsi Calculate Objective Value

Masukan	Proses	Keluaran
Array pop yang menyatakan populasi yang terdiri dari sejumlah chromosome yang terdiri dari *n-binary-string*	Menghitung nilai *fitness* pada masing-masing individu pada populasi	Nilai *fitness* setiap individu pada populasi

Pseudocode 3.6: Fungsi CALCULATE OBJECTIVE VALUE

```
1:  function CALC_OBJECTIVE_VAL(pop)
2:      global train_x, train_y, test_x, test_y
3:      objective_val ← Array []
4:      for p in pop do
5:          hidden_layer ← decode(p)
6:          MLPClassifier((hidden_layer, ), train_x, train_y, params)
7:          training_acc ← score(train_y, predict(train_x))
8:          general_acc ← score(test_y, predict(test_x))
9:          obj_val ← append(e^(training_acc+general_acc))
10:     end for
11:     return obj_val
12: end function
```

3.3.2. Desain Fungsi Seleksi Parents

Fungsi ini merupakan fungsi untuk seleksi *parents* terbaik dari populasi yang akan digunakan dalam proses reproduksi generasi baru selanjutnya. Algoritma yang digunakan adalah *Stochastic Universal Sampling*. Desain fungsi seleksi *parents* terdapat pada *pseudocode* 3.7. Masukan, proses, dan keluaran pada fungsi terdapat pada Tabel 3.7.

Tabel 3.7 Masukan, Proses, dan Keluaran dari Fungsi Select Parents

Masukan	Proses	Keluaran
Array pop yang menyatakan populasi yang terdiri dari sejumlah chromosome yang terdiri dari *n-binary-string*, *Array* $fitness$ yang menyatakan nilai *fitness* pada masing-masing individu, $n\_parents$ yaitu banyak *parents* yang dibutuhkan	Memilih *parents* terbaik pada populasi	*Array parents*, yaitu individu-individu yang terpilih untuk proses reproduksi selanjutnya

Pseudocode 3.7: Fungsi SELECT PARENTS

```
1:  function SELECT_PARENTS(pop, fitness, n_parents)
2:      point_distance ← sum(fitness) /n_parents
3:      start_point ← random.uniform(0, point_distance)
4:      points ← [start_point + i * point_distance] for i in
    range(n_parents)
5:      parents ← Array []
6:      for p in points do i ← 0, sub_sum ← fitness[0]
7:          while sub_sum < p do
8:              i ← i + 1
9:              sub_sum ← sub_sum + fitness[i]
10:         end while
11:         parents ← append(pop[i])
12:     end for
13:     return parents
14: end function
```

3.3.3. Desain Fungsi Crossover

Fungsi ini merupakan fungsi untuk melakukan *crossover* pada tahap reproduksi. Dari sebanyak 8 individu pada populasi, selain kedua *parents* yang terpilih, pada 6 individu dilakukan proses *crossover* antar *parents*. Desain fungsi *crossover* terdapat pada *pseudocode* 3.8. Masukan, proses, dan keluaran pada fungsi terdapat pada Tabel 3.8.

Tabel 3.8 Masukan, Proses, dan Keluaran dari Fungsi Crossover

Masukan	Proses	Keluaran
Array $parents$ individu *parents* yang terdiri dari sejumlah chromosome yang terdiri dari *n-binary-string*, n_criss yaitu banyak individu yang dilakukan proses *crossover*	Melakukan proses *crossover*, proses reproduksi silang antar *parent*	*Array* $off\_crossover$, yaitu individu-individu baru hasil proses reproduksi yang terdapat proses *crossover*

Pseudocode 3.8: Fungsi CROSSOVER

```
1:  function CROSSOVER(parents, n_cross)
2:      off_crossover ← []
3:      while len(off_crossover) < n_cross do
4:          a, b ← random_int(0, len(parents[0] − 1))
5:          new_chr_1 ← parents[0]
6:          new_chr_2 ← parents[1]
7:          new_chr_1[a : b + 1] ← parents[1][a : b + 1]
8:          new_chr_2[a : b + 1] ← parents[0][a : b + 1]
9:          off_crossover ← append(new_chr_1, new_chr_2)
10:     end while
11:     return off_crossover
12: end function
```

3.3.4. Desain Fungsi Mutation

Fungsi ini merupakan fungsi untuk melakukan mutasi pada tahap reproduksi. Pada sebanyak 6 individu *offspring* hasil proses *crossover* dilakukan proses mutasi pada salah satu gen-nya. Desain fungsi mutasi terdapat pada *pseudocode* 3.9. Masukan, proses, dan keluaran pada fungsi terdapat pada Tabel 3.9.

Tabel 3.9 Masukan, Proses, dan Keluaran dari Fungsi MUTATION

Masukan	Proses	Keluaran
Array $offspring$ yang menyatakan populasi pada individu hasil reproduksi dengan *crossover* sebelumnya, n_flip yaitu banyak bit yang diubah atau dilakukan mutasi	Melakukan proses mutasi pada salah satu atau lebih gen pada beberapa individu pada populasi	*Array* $off\_mutation$, yaitu individu-individu baru hasil proses mutasi

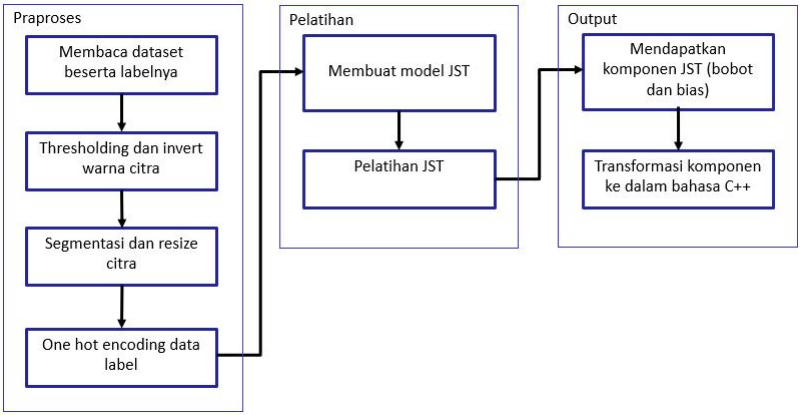
Pseudocode 3.9: Fungsi MUTATION

```
1: function MUTATION(offspring, n_flip)
2:     off_mutation ← offspring
3:     for i in range len(off_spring) do
4:         x ← random(0, n_offspring[0] − 1)
5:         off_mutation[i][x] ← flip_bit()
6:     end for
7:     return off_mutation
8: end function
```

3.4. Desain Pelatihan Bobot Jaringan Saraf Tiruan

Setelah mendapatkan arsitektur terbaik, yaitu banyak neuron terbaik pada *hidden layer* menggunakan optimasi *Genetic Algorithm*, dilakukan proses pelatihan Jaringan Saraf Tiruan untuk mendapatkan bobot yang optimal yang akan digunakan pada proses *testing* pada program yang akan diunggah ke sistem SPOJ pada permasalahan *Hard Image Recognition*.

Gambar 3.5 merupakan diagram proses pelatihan untuk mendapatkan bobot optimal dari Jaringan Saraf Tiruan.



Gambar 3.5 Proses Pelatihan Bobot menggunakan Jaringan Saraf Tiruan

Untuk menjalankan fungsi jaringan saraf tiruan atau *artificial neural network*, penulis menggunakan fungsi MLPCLASSIFIER yang tersedia pada *library* scikit-learn. Pada fungsi ini dibatasi parameter yang digunakan pada ANN, yaitu:

1. *Hidden layer* yang digunakan sebanyak 1
2. Maksimum iterasi sebanyak 1000
3. Fungsi aktivasi yang digunakan adalah *logistic* atau disebut juga *sigmoid*

4. Fungsi optimasi yang digunakan adalah *Adam*
5. *Random State* yang digunakan 42
6. *Learning Rate* sebesar 0.01

pseudocode 3.10 merupakan program utama untuk menjalankan *Artificial Neural Network*. Masukan, proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.10.

Tabel 3.10 Masukan, Proses, dan Keluaran dari Fungsi Artificial Neural Network

Masukan	Proses	Keluaran
Array $data\_x$ yang menyatakan citra dataset yang telah dibaca sistem, $data\_y$ yang menyatakan label dari dataset (0 - 9), n_neuron yang menyatakan banyak *neuron* pada *hidden layer*	Proses Pelatihan Jaringan Saraf Tiruan	Hasil model optimal dari pelatihan

Pseudocode 3.10: Fungsi ARTIFICIAL NEURAL NETWORK

```
1: function ANN(data_x, data_y, n_neuron)
2:     mlp ← MLPClassifier(n_neuron, max_iter, activation, solver, tol, learning_rate_init, random_state)
3:     mlp.fit( data_x, data_y )
4:     get_weights_biases(mlp)
5: end function
```

Setelah dilakukan proses pelatihan pada dataset, bobot dan bias dari hasil pelatihan akan dikeluarkan untuk menjadi parame-

ter pada program *testing* sebagai penyelesaian permasalahan yang akan diunggah ke SPOJ HIR *Hard Image Recognition*. *Pseudocode* 3.11 merupakan fungsi untuk mendapatkan nilai bobot dan bias optimal dari pelatihan. Masukan, proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.11.

Tabel 3.11 Masukan, Proses, dan Keluaran dari Fungsi Mendapatkan Bobot dan Bias

Masukan	Proses	Keluaran
mlp yang menyatakan model *MLPClassifier* yang telah dilatih pada dataset	Proses mendapatkan bobot dan bias optimal	Hasil bobot dan bias yang didapatkan dari pelatihan JST

Pseudocode 3.11: Fungsi Mendapatkan Bobot dan Bias

```
1: function GET_WEIGHTS_BIASES(mlp)
2:     weights ← mlp.coefs_
3:     biases ← mlp.intercepts_
4:     rounded_weights ← [round(weights, 2)]
5:     rounded_biases ← [round(biases, 2)]
6:     Print(rounded_weights)
7:     Print(rounded_biases)
8: end function
```

3.5. Desain Pengujian Jaringan Saraf Tiruan

Setelah melakukan proses pelatihan dan optimasi menggunakan dataset yang telah dibuat, dilakukan pembangunan program pengujian yang digunakan untuk diunggah pada server SPOJ pada permasalahan *Hard Image Recognition*. Program pengujian ini dibangun menggunakan bahasa pemrograman C++.

3.5.1. Desain Fungsi Main

Fungsi akan diawali dengan menerima masukan berupa nilai T yang menyatakan banyaknya kasus uji. Setiap kasus uji diawali dengan sebuah baris dengan dua buah bilangan bulat H dan W yang menyatakan tinggi dan lebar citra. Kemudian diikuti dengan H buah baris dengan masing-masing baris berisi W buah karakter '.' atau 'X' yang merepresentasikan citra.

Setelah menerima citra masukan, fungsi akan melakukan segmentasi untuk mendapatkan bagian yang merupakan *foreground*. Apabila pada saat proses segmentasi bagian yang didapat hanya 1 bagian, maka akan dilakukan proses penghapusan derau coretan. Setelah itu apabila banyak bagian yang didapat lebih dari 6, dilakukan proses penghilangan derau yang lain seperti bagian yang terlalu kecil seperti titik atau yang terlalu besar seperti *frame*. Selain itu dilakukan proses pengecekan apabila bagian yang didapat kurang dari 6, maka dilakukan proses pemisahan suatu bagian agar tidak terdapat digit yang berdempetan. Kemudian setelah mendapatkan kandidat ke-enam digit, dilakukan proses normalisasi ukuran sehingga setiap digit memiliki panjang dan lebar yang sama sebelum masuk ke dalam model Jaringan Saraf Tiruan untuk mendapatkan prediksi digit tersebut.

Pseudocode fungsi Main dapat dilihat pada *pseudocode* 3.13. Daftar nama variabel beserta fungsinya terdapat pada Tabel 3.13. *Struct* dari digit terdapat pada *pseudocode* 3.12 Daftar nama variabel pada *struct* digit beserta fungsinya terdapat pada Tabel 3.12.

Tabel 3.12 Nama dan Fungsi Variabel dalam Struct Digit

Nama Variabel	Fungsi Variabel
$xmin$	Menyimpan posisi kolom paling kiri dari suatu digit
$xmax$	Menyimpan posisi kolom paling kanan dari suatu digit
$ymin$	Menyimpan posisi baris paling kiri dari suatu digit
$ymax$	Menyimpan posisi baris paling kanan dari suatu digit

Pseudocode 3.12: Struct DIGIT

```
1: int xmin
2: int xmax
3: int ymin
4: int ymax
```

Tabel 3.13 Nama dan Fungsi Variabel dalam fungsi Main Pengujian

Nama Variabel	Fungsi Variabel
tc	Menyimpan masukan banyaknya kasus uji
h, w	Menyimpan tinggi dan lebar dari masukan citra
pic	*Array* yang menyimpan karakter '.' atau 'X' dari citra
$meanw, meanh$	Menyimpan rata-rata dari lebar dan tinggi digit-digit
$h1, w1$	Menyimpan tinggi dan lebar awal dari digit
new_w, new_h	Menyimpan tinggi dan lebar baru dari digit
$flatten$	*Array* digit dalam satu dimensi

Pseudocode 3.13: Fungsi MAIN

```
1:  tc ← Input()
2:  for t in range (0, tc) do
3:      h, w ← Input()
4:      pic[i][j] ← Input()
5:      digits ← getDigitCandidate(h, w)
6:      if digits.size == 1 then
7:          removeStruckout(h, w)
8:      end if
9:      totw, toth ← all digits width and height
10:     meanw, meanh ← totw/6, toth/6
11:     if digits.size > 6 then
12:         removeNoise(meanh, meanw)
13:     end if
14:     if digits.size < 6 then
15:         splitOverlappingDigit(toth, totw)
16:     end if
17:     for i in range (0, digits.size) do
18:         temp[0..digits[i].height][0..digits[i].width] ← v[][]
19:         h1 ← digits[i].height
20:         w1 ← digits[i].width
21:         flatten[] ← temp.flatten
22:         resizePixels(h1, w1, new_h, new_w)
23:         loadModel()
24:     end for
25:     printOutput ans
26: end for
```

3.5.2. Desain Fungsi Mendapatkan Kandidat Digit

Fungsi ini digunakan untuk mendapatkan bagian-bagian yang merupakan *foreground* pada citra masukan. Pada fungsi ini akan dilakukan fungsi DFS untuk segmentasi pada citra masukan. Masuk-

an, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.14. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.14.

Tabel 3.14 Masukan, Proses, dan Keluaran dari Fungsi Mendapatkan Kandidat Digit

Masukan	Proses	Keluaran
h yang menyatakan tinggi dari citra dan w yang menyatakan lebar dari citra	Proses mendapatkan kandidat digit atau bagian yang merupakan *foreground* dari citra masukan	*Vector* yang berisi bagian dari citra yang merupakan *foreground*

Pseudocode 3.14: Fungsi Mendapatkan Kandidat Digit

```
1:  function GETDIGITCANDIDATE(h, w)
2:      vector digit_res
3:      memset(v, 0)
4:      for c in range (0, w) do
5:          for r in range (0, h) do
6:              if !v[r][c] and pic[r][c] ==' X' then
7:                  dgt.xmin, dgt.ymin ← 1000
8:                  dgt.xmax, dgt.ymax ← −1
9:                  dfs(r, c, h, w)
10:                 if checkDigit(h, w) then
11:                     digit_res.append(dgt)
12:                 end if
13:             end if
14:         end for
15:     end for
16:     return digit_res
17: end function
```

3.5.3. Desain Fungsi Pengecekan Digit

Fungsi ini digunakan untuk melakukan pengecekan digit pada bagian hasil segmentasi dengan menggunakan tinggi dan lebar bagian tersebut. Apabila tinggi atau lebar kurang dari atau lebih dari *threshold* yang telah ditentukan. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.15. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.15.

Tabel 3.15 Masukan, Proses, dan Keluaran dari Fungsi Pengecekan Digit

Masukan	Proses	Keluaran
h yang menyatakan tinggi dari citra dan w yang menyatakan lebar dari citra	Proses pengecekan apakah bagian yang telah dilakukan proses segmentasi	*Boolean* True atau False apakah bagian tersebut merupakan digit atau bukan

Pseudocode 3.15: Fungsi Pengecekan Digit

```
1:  function CHECKDIGIT(h, w)
2:      if dgt.xmax − dgt.xmin + 1 < 4 then
3:          return False
4:      else if dgt.ymax − dgt.ymin + 1 < 4 then
5:          return False
6:      else if dgt.xmax − dgt.xmin + 1 == w then
7:          return False
8:      else if dgt.xmax − dgt.xmin + 1 == w then
9:          return False
10:     end if
11:      return True
12: end function
```

3.5.4. Desain Fungsi Menghapus Derau Coretan

Fungsi ini digunakan untuk menghapus derau coretan (*struckout noise*) pada citra yang hanya memiliki satu bagian setelah dilakukan segmentasi. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.16. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.16.

Tabel 3.16 Masukan, Proses, dan Keluaran dari Fungsi Menghapus Derau Coretan

Masukan	Proses	Keluaran
h yang menyatakan tinggi dari citra dan w yang menyatakan lebar dari citra	Proses penghapusan derau coretan pada citra yang hanya memiliki satu bagian setelah dilakukan segmentasi	-

Pseudocode 3.16: Fungsi Menghapus Derau Coretan

```
1:  function REMOVESTRUCKOUT(h, w)
2:      for r in range (0, h) do
3:          for c in range (0, w) do
4:              if pic[r][c] ==' X' then
5:                  if nneighbors'X' <= 2 then
6:                      cleaned_pic[r][c] ←'.'
7:                  end if
8:              end if
9:          end for
10:     end for
11:     pic ← cleaned_pic
12:     digits ← getDigitCandidate(h, w)
13: end function
```

3.5.5. Desain Fungsi Menghapus Derau

Fungsi ini digunakan untuk menghapus derau pada bagian yang memiliki nilai lebih besar atau lebih kecil dari rata-rata tinggi dan lebar dari kandidat digit yang didapat. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.17. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.17.

Tabel 3.17 Masukan, Proses, dan Keluaran dari Fungsi Menghapus Derau

Masukan	Proses	Keluaran
$meanh$ dan $meanw$ yang menyatakan rata-rata dari total tinggi dan lebar dari kandidat digit	Proses penghapusan derau pada kandidat yang memiliki tinggi atau lebar yang kurang atau lebih dari *threshold*	-

Pseudocode 3.17: Fungsi Menghapus Derau

```
1:  function REMOVENOISE(meanh, meanw)
2:      for i in range (0, digits.size) do
3:          if digits[i].width + 1 < meanw − 7 then
4:              continue
5:          else if digits[i].width + 7 then
6:              continue
7:          else if digits[i].height > meanh + 7 then
8:              continue
9:          else if digits[i].height > meanh + 7 then
10:             continue
11:         end if
12:         new_digits ←.append(digits[i])
13:     end for
14:     digits ← new_digits
15: end function
```

3.5.6. Desain Fungsi Memisahkan Digit yang Berhimpit

Fungsi ini digunakan untuk memisahkan digit yang berhimpit apabila total kandidat digit yang didapat masih kurang dari 6. Proses dilakukan dengan membagi digit yang memiliki lebar paling besar kemudian dibagi dengan total lebar dibagi 6. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.18. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.18.

Tabel 3.18 Masukan, Proses, dan Keluaran dari Fungsi Memisahkan Digit yang Berhimpit

Masukan	Proses	Keluaran
$toth$ yang menyatakan total tinggi dari kandidat digit dan $totw$ yang menyatakan total lebar dari kandidat digit	Proses pemisahan bagian yang memiliki digit berhimpit, yaitu yang memiliki lebar lebih dari rata-rata	-

3.5.7. Desain Fungsi Normalisasi Ukuran

Fungsi ini digunakan untuk melakukan normalisasi ukuran dari digit sebelum dimasukkan ke dalam fungsi JST. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.19. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.19.

Pseudocode 3.18: Fungsi Memisahkan Digit yang Berhimpit

```
1: function SPLITOVERLAPPINGDIGIT(toth, totw)
2:     sort(digit.width, descending)
3:     meanw, meanh ← totw/6, toth/6
4:     while digits.size < 6 do
5:         n_split ← max(width_digit[i]/meanw, 2)
6:         wi ← width_digit[i]/n_split
7:         for i in range (0, n_split) do
8:             split.xmin ← split.xmax + 1
9:             split.xmax ← min(split.xmin + wi − 1, split.xmax)
10:            split.ymin ← split.ymin
11:            split.ymax ← split.ymax
12:            digits.append(split)
13:        end for
14:    end while
15:    sort(digits)
16: end function
```

Pseudocode 3.19: Fungsi Normalisasi Ukuran

```
1: function RESIZEPIXELS(h1, w1, h2, w2)
2:     memset(resized, 0)
3:     x_ratio ← ((w1 << 16)/w2) + 1
4:     y_ratio ← ((h1 << 16)/h2) + 1
5:     for r in range (0, h2) do
6:         for c in range (0, w2) do
7:             x2 ← ((c ∗ x_ratio) >> 16)
8:             y2 ← ((r ∗ y_ratio) >> 16)
9:             resized[r][c] ← flatten[(y2 ∗ w1) + x2]
10:        end for
11:    end for
12: end function
```

Tabel 3.19 Masukan, Proses, dan Keluaran dari Fungsi Normalisasi Ukuran

Masukan	Proses	Keluaran
$h1$ yang menyatakan tinggi awal dari digit, $w1$ yang menyatakan lebar awal dari digit, $h2$ yang menyatakan tinggi akhir dari digit, $w2$ yang menyatakan lebar akhir dari digit	Proses normalisasi ukuran dari digit	-

3.5.8. Desain Fungsi Model Jaringan Saraf Tiruan

Fungsi ini digunakan untuk menghitung perkalian dengan inputan digit menggunakan model Jaringan Saraf Tiruan dengan bobot dan bias hasil dari proses pelatihan. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.20. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.20.

Tabel 3.20 Masukan, Proses, dan Keluaran dari Fungsi Model Jaringan Saraf Tiruan

Masukan	Proses	Keluaran
Digit yang telah dilakukan normalisasi ukuran	Proses perhitungan luaran dari model Jaringan Saraf Tiruan	-

Pseudocode 3.20: Fungsi Model Jaringan Saraf Tiruan

```
1: function LOADMODEL(h1, w1, h2, w2)
2:     w1[new_w*new_h][size_hdn_layer], w2[size_hdn_layer][10] ← weights from pretrained model
3:     b1[size_hdn_layer], b2[10] ← biases from pretrained model
4:     for h in range (0, size_hdn_layer) do
5:         for i in range (0, new_h) do
6:             for j in range (0, new_w) do
7:                 tmp ← tmp + resized[i][j] * w1[i][j]
8:             end for
9:         end for
10:        hdn_lyr1[h] ← actFunc(tmp + b1[bi])
11:    end for
12:    for i in range (0, 10) do
13:        for r in range (0, size_hdn_layer) do
14:            for j in range (0, new_w) do
15:                tmp ← tmp + hdn_lyr1[r] * w2[i][j]
16:            end for
17:        end for
18:        output[i] ← actFunc(tmp + b2[bi])
19:    end for
20:    predict(output, 10)
21: end function
```

3.5.9. Desain Fungsi Aktivasi

Fungsi ini digunakan untuk menghitung luaran setelah dilakukan aktivasi yang bertujuan untuk menormalkan luaran hasil perkalian yang bisa jadi cukup besar menggunakan fungsi aktivasi Sigmoid. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.21. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.21.

Tabel 3.21 Masukan, Proses, dan Keluaran dari Fungsi Aktivasi

Masukan	Proses	Keluaran
x yaitu nilai sebelum dilakukan proses aktivasi	Proses perhitungan aktivasi luaran JST menggunakan fungsi aktivasi Sigmoid	Luaran yang normal setelah dilakukan fungsi aktivasi

Pseudocode 3.21: Fungsi Aktivasi

```
1: function ACTFUNC(x)
2:     return 1/(1 + exp(−x))
3: end function
```

3.5.10. Desain Fungsi Prediksi Kelas Digit

Fungsi ini digunakan untuk menghitung prediksi kelas dari digit setelah melakukan perhitungan perkalian antar bobot dan bias. Masukan, Proses, dan keluaran dari fungsi ini tercantum pada Tabel 3.22. *Pseudocode* dari fungsi ini terdapat pada *pseudocode* 3.22.

Tabel 3.22 Masukan, Proses, dan Keluaran dari Fungsi Prediksi Kelas Digit

Masukan	Proses	Keluaran
$output$ yaitu *Array* hasil perkalian setiap bobot dengan masukan digit, $size$ banyaknya kelas yaitu 10	Perhitungan untuk menentukan kelas dari digit	Kelas dari digit

Pseudocode 3.22: Fungsi Prediksi Kelas Digit

```
1: function PREDICT(x)
2:     for i in range (0, size) do
3:         if output[i] > tmp then
4:             tmp ← output[i]
5:             dgt_pred ← i
6:         end if
7:     end for
8:     ans.append(dgt_pred)
9: end function
```

BAB IV
IMPLEMENTASI

4.1. Lingkungan implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

4.1.1. Program Pembuatan Dataset, Pelatihan Fungsi JST, dan Optimasi GA

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700 CPU @ 3.4GHz (4 CPUs), 3.4GHz
   (b) Random Access Memory 24576MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Jupyter Notebook
   (c) Bahasa Pemrograman Python 3.6
   (d) MinGW 64-bit

4.1.2. Program Pengujian Fungsi JST

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700 CPU @ 3.4GHz (4 CPUs), 3.4GHz
   (b) Random Access Memory 24576MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) MinGW 64-bit

4.2. Implementasi Program Pembuatan Dataset

Subbab ini menjelaskan implementasi program pembuatan dataset yang digunakan sebagai data latih untuk model Jaringan Saraf Tiruan yang dibangun. Program ini ditulis menggunakan bahasa pemrograman Python.

4.2.1. Library yang diperlukan

Program ini menggunakan beberapa *library* seperti yang ditunjukkan pada kode sumber 4.1. Penjelasan fungsi dari *library* yang digunakan terdapat pada Tabel 4.1.

```
1   import numpy as np
2   import os
3   import glob
4
5   import cv2
6   import PIL
7   from PIL import ImageFont
8   from PIL import Image
9   from PIL import ImageDraw
```

Kode Sumber 4.1 *Library* Program Pembuatan Dataset

Tabel 4.1 Nama dan Penjelasan *library* dalam Pembuatan Dataset

Nama Library	Penjelasan
numpy	library dasar untuk komputasi ilmiah yang berhubungan dengan *array* dan *matrix*
os	untuk melakukan sesuatu pada direktori, seperti masuk ke dalam suatu folder
glob	untuk membuat list dari file dengan pola yang diberikan
cv2	untuk membuat dan memanipulasi gambar
PIL	serupa dengan opencv, digunakan untuk membuat dan memanipulasi gambar

4.2.2. Implementasi Pembuatan Dataset dan Penambahan Dataset HIR

Subbab ini menjabarkan pembuatan dataset dengan menggunakan *font* yang telah diunduh. Sebuah *blank image* berwarna putih sebesar 32x32 pixel dibuat untuk menjadi *background* dari citra digit. Kemudian citra digit dari 0-9 digambar pada *background* tersebut dengan 3 ukuran yang berbeda. Implementasi tahap ini dapat dilihat pada kode sumber 4.2.

```
1   path = os.path.join(os.getcwd(), "Font",
        "*.ttf")
2   files = glob.glob(path)
3
4   fonts = {}
5   size = [24, 20, 16]
6
7   for f in files:
8     fonts[f.split('/')[-1][:-4]] = f
9
10  for i in range(0, 10):
11    for f in fonts:
12      for s in size:
13        file_out = "dataset/train/" + str(i) +
              "_" + f + "_" + str(s) + ".png"
14        draw_digit(file_out, str(i), fonts[f], s)
```

Kode Sumber 4.2 Tahap Pembuatan Dataset

Selain membuat dataset menggunakan font yang telah diunduh, penulis juga menggunakan dataset yang terdapat pada situs diskusi SPOJ [2]. Dengan menggunakan segmentasi secara manual pada bagian-bagian yang merupakan digit pada masing-masing citra. Posisi-posisi titik kiri-atas, titik kanan-atas, titik kiri-bawah dan titik kanan-bawah dari keenam digit dimasukkan secara manual. Implementasi tahap ini dapat dilihat pada kode sumber 4.3.

```
 1   path = os.path.join(os.getcwd(), "dataset_hir",
         "*.jpg")
 2   files = glob.glob(path)
 3
 4   img = cv2.imread(files[0], cv2.IMREAD_GRAYSCALE)
 5   ret, img_th = cv2.threshold(img, 127, 255,
         cv2.THRESH_BINARY_INV)
 6
 7   img_th = img_th[7:34, 8:115]
 8
 9   dgt_list = [
10     [1, 20, 2, 13],
11     [1, 20, 19, 32],
12     [1, 20, 36, 49],
13     [1, 20, 53, 64],
14     [1, 20, 70, 83],
15     [1, 20, 87, 100]
16   ]
17
18   i = 0
19   for d in dgt_list:
20     filename = "dataset/test/" + "hir_001_" +
           str(i) + ".png"
21     hir_digit(img_th, d, filename)
22
23     i+=1
```

Kode Sumber 4.3 Tahap Penambahan Dataset

4.2.3. Implementasi Fungsi Menggambar Digit

Subbab ini menjabarkan fungsi untuk menggambar digit dari 0-9 dengan *font* dan ukuran yang telah diberikan. Implementasi tahap ini dapat dilihat pada kode sumber 4.4.

```
1  def draw_digit(file_out, digit, font_type, size):
2    img = Image.new("RGBA", (32,32), (255,255,255))
3
4    font = ImageFont.truetype(font_type, size)
5    w, h = font.getsize(digit)
6
7    draw = ImageDraw.Draw(img)
8    draw.text(((32-w)/2, (32-h)/2), digit,
         font=font, fill="black")
9
10   img.save(file_out)
```

Kode Sumber 4.4 Fungsi Menggambar Digit Menggunakan Font

4.2.4. Implementasi Fungsi Segmentasi Digit HIR

Subbab ini menjabarkan fungsi untuk melakukan segmentasi pada citra HIR sesuai dengan posisi digit yang telah diberikan. Pada fungsi ini terdapat proses untuk mendapatkan area terbesar dari suatu daerah yang kemudian akan disalin ke *blank image* dan disimpan. Implementasi tahap ini dapat dilihat pada kode sumber 4.5 dan 4.6.

```
1      new_dgt = np.zeros((32, 32), np.uint8)
2      new_dgt[:, :] = 255
3
4      dgt = img[pos[0]:pos[1], pos[2]:pos[3]]
5      ret, dgt = cv2.threshold(dgt, 127, 255,
           cv2.THRESH_BINARY_INV)
```

Kode Sumber 4.5 Fungsi Segmentasi Digit HIR (1)

```
 6      dgt_resized = cv2.resize(dgt, (20, 30),
            interpolation=cv2.INTER_NEAREST)
 7      new_dgt[1:-1, 6:-6] = dgt_resized
 8
 9      ret, temp = cv2.threshold(new_dgt, 127, 255,
            cv2.THRESH_BINARY_INV)
10      ret, labels = cv2.connectedComponents(temp)
11      labels_props = measure.regionprops(labels)
12
13      max_area = 0
14      dgt_index = 0
15
16      for label_props in labels_props:
17          if label_props.area > max_area:
18              max_area = label_props.area
19              dgt_index =
                    labels_props.index(label_props)
20
21      dgt_mask = labels == dgt_index+1
22      new_dgt[dgt_mask==False] = 255
23
24      cv2.imwrite(filename, new_dgt)
```

Kode Sumber 4.6 Fungsi Segmentasi Digit HIR (2)

4.3. Implementasi Program Pelatihan dan Optimasi JST menggunakan GA

Subbab ini menjelaskan implementasi program pelatihan dan optimasi arsitektur Jaringan Saraf Tiruan menggunakan *Genetic Algorithm*. Parameter yang akan dioptimasi adalah jumlah *neuron* terbaik pada *hidden layer*. Program ini ditulis menggunakan bahasa pemrograman Python.

4.3.1. Library yang diperlukan

Program ini menggunakan beberapa *library* seperti yang ditunjukkan pada kode sumber 4.7. Penjelasan fungsi dari *library*

yang digunakan terdapat pada Tabel 4.2.

```
1  import numpy as np
2  import os
3  import glob
4
5  import cv2
6
7  from keras.utils import np_utils
8  from sklearn.neural_network import MLPClassifier
9  from sklearn.metrics import accuracy_score
```

Kode Sumber 4.7 *Library* Program Pelatihan dan Optimasi JST menggunakan GA

Tabel 4.2 Nama dan Penjelasan *library* dalam Program Pelatihan dan Optimasi JST menggunakan GA

Nama Library	Penjelasan
numpy	library dasar untuk komputasi ilmiah yang berhubungan dengan *array* dan *matrix*
os	untuk melakukan sesuatu pada direktori, seperti masuk ke dalam suatu folder
glob	untuk membuat list dari file dengan pola yang diberikan
cv2	untuk membuat, membaca, dan memanipulasi gambar
np_utils	untuk melakukan proses *one hot encoding* pada data label
MLPClassifier	untuk memanggil fungsi pembuatan dan pelatihan JST

4.3.2. Implementasi Fungsi Genetic Algorithm

Subbab ini menjabarkan program utama untuk menjalankan proses optimasi arsitektur JST menggunakan GA. Proses GA dimu-

lai dengan menentukan populasi awal yang terdiri dari 8 kromosom yang tersusun atas 6-*binary string* yang menyatakan banyaknya *neuron* pada *hidden layer*. Implementasi tahap ini dapat dilihat pada kode sumber 4.8 dan 4.9.

```
1   pop_size = 8
2   n_gene = 6
3   n_generation = 300
4   n_parents = 2
5
6   population = []
7
8   for i in range(pop_size):
9     population.append([random.randint(0, 1) for x
          in range(n_gene)])
10
11  for generation in range(n_generation):
12    print("Generation: ", generation)
13
14    objective_val = calc_objective_val(population)
15    idx = objective_val.index(max(objective_val))
16
17    hidden_layer = 0
18    for bit in population[idx]:
19      hidden_layer = (hidden_layer << 1) | bit
20    hidden_layer = max(1, hidden_layer)
21
22    print("Best n neurons for hidden layer: ",
          hidden_layer)
23    print("Best fitness: ", objective_val[idx])
24
25    if hidden_layer != best_n:
26        best_n = hidden_layer
27        cntr = 0
28    cntr += 1
29
30    if cntr == 5:
31        break
```

Kode Sumber 4.8 Program Utama Fungsi Optimasi GA (1)

```
32        parents = select_parents(population,
              objective_val, n_parents)
33        offspring_crossover = crossover(parents,
              (pop_size - n_parents))
34
35        n_flip = 1
36        offspring_mutation =
              mutation(offspring_crossover, n_flip)
37
38        population[:n_parents] = parents
39        population[n_parents:] = offspring_mutation
```

Kode Sumber 4.9 Program Utama Fungsi Optimasi GA (2)

4.3.3. Implementasi Fungsi Menghitung Nilai Objektif

Subbab ini menjabarkan program untuk menghitung nilai objektif atau nilai *fitness* dari optimasi GA pada JST. Fungsi objektif yang digunakan adalah memaksimalkan nilai akurasi dari data *training* dan data *testing*. Implementasi tahap ini dapat dilihat pada kode sumber 4.10 dan 4.11.

```
 1 def calc_objective_val(pop):
 2   global train_x, train_y, test_x, test_y
 3
 4   objective_val = []
 5   for p in pop:
 6     hidden_layer = 0
 7     for bit in p:
 8       hidden_layer = (hidden_layer << 1) | bit
 9     hidden_layer = max(1, hidden_layer)
10
11     mlp = MLPClassifier (hidden_layer_sizes =
           (hidden_layer, ),max_iter=1000,
           alpha=1e-4, activation='logistic',
           solver='adam', verbose=0, tol=1e-4,
           random_state=42, learning_rate_init=.01,
           warm_start=True)
```

Kode Sumber 4.10 Tahap Menghitung Nilai Objektif (1)

```
12      mlp.fit(train_x, train_y)
13
14      training_acc = mlp.score(train_x, train_y)*100
15      generalization_acc = mlp.score(test_x,
            test_y)*100
16
17      objective_val.append(np.exp(training_acc +
            generalization_acc))
18
19    return objective_val
```

Kode Sumber 4.11 Tahap Menghitung Nilai Objektif (2)

4.3.4. Implementasi Fungsi Seleksi Parents

Subbab ini menjabarkan program untuk melakukan proses seleksi *parents* terbaik dari populasi yang akan digunakan dalam proses reproduksi generasi baru selanjutnya. Algoritma yang digunakan adalah *Stochastic Universal Sampling*. Implementasi tahap ini dapat dilihat pada kode sumber 4.12 dan 4.13.

```
1 def select_parents(pop, fitness, n_parents):
2   total_fitness = sum(fitness)
3   point_distance = total_fitness/n_parents
4
5   start_point = random.uniform(0, point_distance)
6   points = [start_point + i*point_distance for i
        in range(n_parents)]
7
8   parents = []
```

Kode Sumber 4.12 Tahap Proses Seleksi Parents (1)

```
 9   for p in points:
10     i = 0
11     sub_sum = fitness[0]
12
13     while sub_sum < p:
14       i+=1
15       sub_sum += fitness[i]
16
17     parents.append(pop[i])
18
19   return parents
```

Kode Sumber 4.13 Tahap Proses Seleksi Parents (2)

4.3.5. Implementasi Fungsi Crossover

Subbab ini menjabarkan program untuk melakukan proses *crossover* pada tahap reproduksi. Dari sebanyak 32 indvidu pada populasi, selain kedua *parents* yang terpilih, pada 20 individu dilakukan proses *crossover* antar *parents*, sementara 12 individu yang tersisa tidak dilakukan proses ini. Implementasi tahap ini dapat dilihat pada kode sumber 4.14 dan 4.15.

```
 1 def crossover(parents, n_cross):
 2   off_crossover = []
 3   i=0
 4   while(len(off_crossover) < n_cross):
 5     i+=1
 6     p1_idx = i%len(parents)
 7     p2_idx = (i+1)%len(parents)
 8     hit_a = random.randint(0, len(parents[0])-1)
 9     hit_b = random.randint(0, len(parents[0])-1)
10     start_point = min(hit_a, hit_b)
11     end_point = max(hit_a, hit_b)
```

Kode Sumber 4.14 Tahap Proses Crossover (1)

```
12      new_chromosome_1 = deepcopy(parents[p1_idx])
13      new_chromosome_1[hit_a:hit_b+1] =
            deepcopy(parents[p2_idx][hit_a:hit_b+1])
14      new_chromosome_2 = deepcopy(parents[p2_idx])
15      new_chromosome_2[hit_a:hit_b+1] =
            deepcopy(parents[p1_idx][hit_a:hit_b+1])
16
17      off_crossover.append(new_chromosome_1)
18      off_crossover.append(new_chromosome_2)
19  return off_crossover
```

Kode Sumber 4.15 Tahap Proses Crossover (2)

4.3.6. Implementasi Fungsi Mutasi

Subbab ini menjabarkan program untuk melakukan proses mutasi pada tahap reproduksi. Dari sebanyak 32 individu pada populasi, pada 12 individu dilakukan proses mutasi pada salah satu gen. Implementasi tahap ini dapat dilihat pada kode sumber 4.16.

```
 1 def mutation(offspring, n_flip):
 2   off_mutation = deepcopy(offspring)
 3   for idx in range(len(offspring)):
 4     x = random.randint(0, len(offspring[0])-1)
 5
 6     if off_mutation[idx][x] == 1:
 7       off_mutation[idx][x] = 0
 8     else:
 9       off_mutation[idx][x] = 1
10
11 return off_mutation
```

Kode Sumber 4.16 Tahap Proses Mutasi

4.4. Implementasi Program Pelatihan Bobot Jaringan Saraf Tiruan

Subbab ini menjelaskan implementasi program pelatihan Bobot Jaringan Saraf Tiruan menggunakan arsitektur banyak *neuron* yang merupakan hasil optimasi menggunakan *Genetic Algorithm*. Proses pelatihan JST dilakukan untuk mendapatkan bobot yang optimal yang akan digunakan pada proses *testing* pada program yang akan diunggah ke sistem SPOJ pada permasalahan *Hard Image Recognition*. Program ini ditulis menggunakan bahasa pemrograman Python.

4.4.1. Library yang diperlukan

Program ini menggunakan beberapa *library* seperti yang ditunjukkan pada kode sumber 4.17. Penjelasan fungsi dari *library* yang digunakan terdapat pada Tabel 4.3.

```
1   import numpy as np
2   import os
3   import glob
4   import cv2
5
6   from keras.utils import np_utils
7   from sklearn.neural_network import MLPClassifier
```

Kode Sumber 4.17 *Library* Program Pelatihan Bobot Jaringan Saraf Tiruan

Tabel 4.3 Nama dan Penjelasan *library* dalam program Pelatihan Bobot Jaringan Saraf Tiruan

Nama Library	Penjelasan
numpy	library dasar untuk komputasi ilmiah yang berhubungan dengan *array* dan *matrix*
os	untuk melakukan sesuatu pada direktori, seperti masuk ke dalam suatu folder
glob	untuk membuat list dari file dengan pola yang diberikan
np_utils	untuk melakukan proses *one hot encoding* pada data label
MLPClassifier	untuk memanggil fungsi pembuatan dan pelatihan JST

4.4.2. Implementasi Fungsi Jaringan Saraf Tiruan

Subbab ini menjabarkan fungsi Jaringan Saraf Tiruan. Proses ini menggunakan fungsi MLPClassifier yang tersedia pada *library* scikit-learn. Pada fungsi ini dibatasi parameter yang digunakan pada ANN, sesuai pada subbab 3.4. Implementasi tahap ini dapat dilihat pada kode sumber 4.18.

```
1 def ann(data_x, data_y, n_neuron):
2   mlp =
        MLPClassifier(hidden_layer_sizes=(n_neuron,
        ), max_iter=1000, alpha=1e-4,
        activation='logistic', solver='adam',
        verbose=2, tol=1e-4, random_state=42,
        learning_rate_init=.01)
3
4   mlp.fit(data_x, data_y)
5   get_weights_biases(mlp)
```

Kode Sumber 4.18 Program Fungsi Jaringan Saraf Tiruan

4.4.3. Implementasi Fungsi Mendapatkan Bobot dan Bias

Subbab ini menjabarkan fungsi untuk mendapatkan nilai bobot dan bias optimal dari pelatihan jaringan saraf tiruan. Bobot dan bias yang didapatkan akan dikeluarkan untuk menjadi parameter pada program *testing* sebagai penyelesaian permasalahan yang akan diunggah ke SPOJ *Hard Image Recognition* (HIR). Implementasi tahap ini dapat dilihat pada kode sumber 4.19

```
def get_weights_biases(mlp):
  weights = mlp.coefs_
  biases = mlp.intercepts_

  rounded_weights = [np.round(w, 2) for w in
      weights]
  rounded_biases = [np.round(b, 2) for b in
      biases]

  for i in rounded_weights[0]:
    print("{", end='')
    for j in range(len(i)-1):
      print(i[j], end=', ')
    print(i[len(i)-1], "}, ")

  for i in rounded_weights[1]:
    print("{", end='')
    for j in range(len(i)-1):
      print(i[j], end=', ')
    print(i[len(i)-1], "}, ")
```

Kode Sumber 4.19 Program Fungsi Mendapatkan Bobot dan Bias

4.5. Implementasi Program Pengujian Jaringan Saraf Tiruan

Subbab ini menjelaskan implementasi program pengujian jaringan saraf tiruan dengan menggunakan arsitektur yang sama dengan arsitektur yang digunakan pada proses pelatihan dan menggunakan bobot dan bias optimal yang didapatkan dari hasil pelatihan. Program ini ditulis menggunakan bahasa pemrograman C++.

4.5.1. Penggunakan Library, Variabel Global, dan Struct

Program ini menggunakan beberapa *library* seperti yang ditunjukkan pada kode sumber 4.20. Penjelasan fungsi dari *library* yang digunakan terdapat pada Tabel 4.4.

```
1   #include<iostream>
2   #include<vector>
3   #include<cmath>
4
5   using namespacestd;
```

Kode Sumber 4.20 *Library* Program Pengujian Jaringan Saraf Tiruan

Tabel 4.4 Nama dan Penjelasan *library* dalam program Pengujian Jaringan Saraf Tiruan

Nama Library	Penjelasan
iostream	sebagai standar input output operasi yang digunakan oleh bahasa C++
vector	library standar yang digunakan untuk mengaplikasikan *vector*
cmath	merupakan file header yang berfungsi untuk operasi matematika

Pada program ini terdapat beberapa variabel global yang digunakan yang ditunjukkan pada kode sumber 4.21. Penjelasan fungsi dari variabel global yang digunakan terdapat pada Tabel 4.5.

```
1   char pic[255][255];
2   int v[255][255] = {0}, flatten[65025];
3   digit dgt;
4   vector<digit> digits, new_digits;
5   vector<int> ans;
6   int new_w = 16, new_h = 24, size_hdn_layer = 60;
7   double resized[new_h][new_w] = {0.0};
```

Kode Sumber 4.21 Variabel Global pada Program Pengujian Jaringan Saraf Tiruan

Tabel 4.5 Nama dan Penjelasan Variabel Global dalam Program Pengujian Jaringan Saraf Tiruan

Nama Variabel	Penjelasan
pic	untuk menyimpan citra masukan yang berupa karakter 'X' dan '.'
v	untuk menyimpan *state* telah dikunjungi (*visited*) dari fungsi DFS
flatten	untuk menyimpan transformasi dari citra 2d menjadi 1d
dgt	untuk menyimpan elemen-elemen pada 1 digit
digits	untuk menyimpan elemen-elemen digit yang akan diprediksi
new_digits	untuk menyimpan *temporary* elemen-elemen digit sementara
ans	untuk menyimpan hasil prediksi angka
new_w, new_h	untuk menyimpan lebar dan tinggi baru setelah normalisasi ukuran
size_hidden_layer	untuk menyimpan banyak *neuron* pada *hidden layer*
resized	untuk menyimpan citra setelah dilakukan normalisasi ukuran

Pada program ini juga menggunakan *Struct* untuk menyimpan posisi dari digit. Banyak variabel yang digunakan pada *struct* digit sebanyak 4 variabel seperti yang ditunjukkan pada kode sumber 4.22. Penjelasan fungsi dari *struct* digit terdapat pada Tabel 4.6.

```
1   struct digit {
2     int xmin, xmax, ymin, ymax;
3   };
```

Kode Sumber 4.22 Struct Digit pada Program Pengujian Jaringan Saraf Tiruan

Tabel 4.6 Nama dan Penjelasan Variabel pada Struct Digit dalam Program Pengujian JST

Nama Variabel	Penjelasan
xmin	Menyimpan posisi kolom paling kiri pada digit
xmax	Menyimpan posisi kolom paling kanan pada digit
ymin	Menyimpan posisi baris paling atas pada digit
ymax	Menyimpan posisi baris paling bawah pada digit

4.5.2. Implementasi Fungsi Utama pada Program Pengujian JST

Subbab ini menjabarkan fungsi utama pada program pengujian JST. Program diawali dengan menerima masukan berupa banyak kasus uji dan citra masukan pada setiap kasus uji. Pada setiap kasus uji akan dilakukan proses untuk mendapatkan keenam digit yang terdapat pada citra. Implementasi tahap ini dapat dilihat pada kode sumber 4.23.

```
 1 int main() {
 2   int tc, h, w;
 3   cin >> tc;
 4   for(int t = 0; t < tc; t++){
 5     cin >> h >> w;
 6     float totw, toth, meanw, meanh = 0.0;
 7     for(int i = 0; i < h; i++)
 8       cin >> pic[i];
 9
10     digits = getDigitCandidate(h, w);
11     if(digits.size() == 1) removeStruckout(h, w);
12     if(digits.size() > 6) removeNoise(meanh,
           meanw);
13     if(digits.size() > 0 && digits.size() < 6)
14       splitOverlappingDigit(toth, totw);
15
16     for(int i = 0; i < digits.size(); i++){
17       int temp[255][255] = {0};
18       for(int r = digits[i].ymin; r <=
             digits[i].ymax; r++){
19         for(int c = digits[i].xmin; c <=
               digits[i].xmax; c++){
20           temp[r - digits[i].ymin][c -
                 digits[i].xmin] = v[r][c];
21         }
22       }
23       int h1 = digits[i].ymax - digits[i].ymin +1;
24       int w1 = digits[i].xmax - digits[i].xmin +1;
25       for(int r = 0; r < h1; r++)
26         for(int c = 0; c < w1; c++)
27           flatten[(r*w1) + c] = temp[r][c];
28       resizePixels(h1, w1, new_h, new_w);
29       loadModel();
30     }
31     for(int i = 0; i < ans.size(); i++)
32       printf("%d", ans[i]);
33   }
34 }
```

Kode Sumber 4.23 Program Fungsi Utama pada Pengujian JST

4.5.3. Implementasi Fungsi Mendapatkan Kandidat Digit

Subbab ini menjabarkan fungsi untuk mendapatkan bagian-bagian yang merupakan *foreground* pada citra masukan. Pada fungsi ini akan dilakukan fungsi DFS untuk melakukan proses segmentasi pada citra masukan. Keluaran dari fungsi ini berupa *vector* yang berisi *struct* digit. Implementasi tahap ini dapat dilihat pada kode sumber 4.24.

```
vector<digit> getDigitCandidate(int h, int w){
  vector<digit> digit_res;

  for(int r = 0; r < h; r++){
    for(int c = 0; c < w; c++){
      v[r][c] = 0;
    }
  }

  for(int c = 0; c < w; c++){
    for(int r = 0; r < h; r++){
      if(!v[r][c] && pic[r][c] == 'X'){
        dgt.xmin = 1000;
        dgt.xmax = -1;
        dgt.ymin = 1000;
        dgt.ymax = -1;

        dfs(r, c, h, w);

        if(checkDigit(h, w))
            digit_res.push_back(dgt);
      }
    }
  }

  return digit_res;
}
```

Kode Sumber 4.24 Program Fungsi Mendapatkan Kandidat Digit

4.5.4. Implementasi Fungsi Pengecekan Digit

Subbab ini menjabarkan fungsi untuk melakukan pengecekan digit pada bagian hasil segmentasi dengan menggunakan tinggi dan lebar bagian tersebut. Pada program ini digunakan *threshold* sebesar 4. Implementasi tahap ini dapat dilihat pada kode sumber 4.25.

```
1 bool checkDigit(int h, int w){
2   if((dgt.xmax - dgt.xmin + 1 < 4) || (dgt.ymax -
        dgt.ymin + 1 < 4)) return false;
3   else if( (dgt.xmax - dgt.xmin + 1 == w) ||
        (dgt.ymax - dgt.ymin + 1 == h)) return
        false;
4
5   return true;
6 }
```

Kode Sumber 4.25 Program Fungsi Pengecekan Digit

4.5.5. Implementasi Fungsi Menghapus Derau Coretan

Subbab ini menjabarkan fungsi untuk menghapus derau coretan pada citra. Implementasi tahap ini dapat dilihat pada kode sumber 4.26 dan 4.27.

```
 1 void removeStruckout(int h, int w){
 2   const int dx[] = {0, 1, 0, -1, -1, 1, 1, -1};
 3   const int dy[] = {1, 0, -1, 0, 1, -1, +1, -1};
 4   char cleaned_pic[h][w];
 5
 6   for(int r = 0; r < h; r++){
 7     for(int c = 0; c < w; c++){
 8       if(pic[r][c] == 'X'){
 9         int cnt = 0;
10         for(int i = 0; i < 8; i++){
11           if(pic[r + dy[i]][c + dx[i]] == 'X')
12             cnt+=1;
13         }
```

Kode Sumber 4.26 Program Fungsi Menghapus Derau Coretan (1)

```
14          if(cnt <= 2) cleaned_pic[r][c] = '.';
15          else cleaned_pic[r][c] = 'X';
16        }
17        else cleaned_pic[r][c] = '.';
18      }
19    }
20
21    for(int r = 0; r < h; r++){
22      for(int c = 0; c < w; c++){
23        pic[r][c] = cleaned_pic[r][c];
24      }
25    }
26
27    digits = getDigitCandidate(h, w);
28  }
```

Kode Sumber 4.27 Program Fungsi Menghapus Derau Coretan (2)

4.5.6. Implementasi Fungsi Menghapus Derau

Subbab ini menjabarkan fungsi untuk menghapus derau pada bagian yang memiliki nilai lebih besar atau lebih kecil dari rata-rata tinggi dan lebar dari kandidat digit yang didapat. Implementasi tahap ini dapat dilihat pada kode sumber 4.28.

```
 1 void removeNoise(float meanh, float meanw){
 2   for(int i = 0; i < digits.size(); i++){
 3     if(
 4     digits[i].xmax-digits[i].xmin+1 < meanw-7 ||
 5     digits[i].xmax-digits[i].xmin+1 > meanw+7 ||
 6     digits[i].ymax-digits[i].ymin+1 < meanh-7 ||
 7     digits[i].ymax-digits[i].ymin+1 > meanh+7){
 8       continue;
 9     }
10     new_digits.push_back(digits[i]);
11   }
12   digits = new_digits; new_digits.clear();
13 }
```

Kode Sumber 4.28 Program Fungsi Menghapus Derau

4.5.7. Implementasi Fungsi Memisahkan Digit yang Berhimpit

Subbab ini menjabarkan fungsi untuk memisahkan digit yang berhimpit apabila total kandidat digit yang didapat masih kurang dari 6, karena kemungkinan terdapat dua atau lebih digit yang berhimpit. Implementasi tahap ini dapat dilihat pada kode sumber 4.29 dan 4.30.

```
1  void splitOverlappingDigit(float toth, float
       totw){
2    vector<pair<int, int> > width;
3
4    for(int i = 0; i < digits.size(); i++){
5      width.push_back(make_pair(digits[i].xmax -
           digits[i].xmin + 1, i));
6    }
7
8    sort(width.begin(), width.end(),
         greater<pair<int, int> >());
9
10   float meanw = totw / 6;
11   float meanh = toth / 6;
12
13   int hit = 0;
14
15   while(digits.size() < 6){
16     int n_split = max(int(round(width[hit].fi /
           meanw)), 2);
17     int wi = ceil(width[hit].fi / n_split);
18     int id = width[hit].se;
19
20     digit temp = digits[id];
21     digit split;
22
23     split.xmin = temp.xmin;
24     split.xmax = temp.xmin + wi -1;
25     split.ymin = temp.ymin;
26     split.ymax = temp.ymax;
```

Kode Sumber 4.29 Program Fungsi Memisahkan Digit yang Berhimpit (1)

```
27      digits[id] = split;
28      for(int i = 1; i < n_split; i++){
29        split.xmin = split.xmax + 1;
30        split.xmax = min(split.xmin + wi -1,
              temp.xmax);
31        split.ymin = temp.ymin;
32        split.ymax = temp.ymax;
33
34        digits.push_back(split);
35      }
36
37      hit++;
38    }
39
40    sort(digits.begin(), digits.end(),
          &digit_sorter);
41  }
```

Kode Sumber 4.30 Program Fungsi Memisahkan Digit yang Berhimpit (2)

4.5.8. Implementasi Fungsi Normalisasi Ukuran

Subbab ini menjabarkan fungsi untuk melakukan normalisasi ukuran dari digit sebelum menjadi masukan pada fungsi JST. Fungsi ini menggunakan algoritma *nearest neighbor* untuk melakukan perubahan ukuran. Implementasi tahap ini dapat dilihat pada kode sumber 4.31.

```
 1 void resizePixels(int h1, int w1, int h2, int w2)
       {
 2   for(int r = 0; r < new_h; r++){
 3     for(int c = 0; c < new_w; c++){
 4       resized[r][c] = 0.0;
 5     }
 6   }
 7
 8   int x_ratio = (int)((w1<<16)/w2) +1;
 9   int y_ratio = (int)((h1<<16)/h2) +1;
10
11   int x2, y2;
12   for(int r = 0; r < h2; r++) {
13     for(int c = 0; c < w2; c++) {
14       x2 = ((c*x_ratio)>>16);
15       y2 = ((r*y_ratio)>>16);
16       resized[r][c] = double(flatten[(y2*w1)+x2]);
17     }
18   }
19 }
```

Kode Sumber 4.31 Program Fungsi Normalisasi Ukuran

4.5.9. Implementasi Fungsi Model Jaringan Saraf Tiruan

Subbab ini menjabarkan fungsi Jaringan Saraf Tiruan untuk menghitung luaran menggunakan bobot dan bias hasil dari proses pelatihan. Nilai bobot dan bias yang didapat dari hasil uji coba sangat banyak. Maka dari itu, untuk nilai bobot dan bias akan dimasukkan pada lampiran pada buku ini. Implementasi tahap ini dapat dilihat pada kode sumber 4.32.

```
 1 void loadModel(){
 2   double w1[new_w*new_h][size_hdn_layer]={...};
 3   double w2[size_hdn_layer][10]={...};
 4   double b1[size_hdn_layer]={...};
 5   double b2[10]={...};
 6   double hdn_lyr1[size_hdn_layer + 5] = {0.0};
 7   double output[10] = {0.0};
 8   double tmp, activated;
 9
10   int wi, wj = 0, bi = 0;
11   for(int h = 0; h < size_hdn_layer; h++){
12     tmp = 0.0; wi = 0;
13     for(int i = 0; i < new_h; i++){
14       for(int j = 0; j < new_w; j++){
15         tmp += (resized[i][j] * w1[wi][wj]);
16         wi += 1;
17       }
18     }
19     activated = actFunc(tmp + b1[bi]);
20     hdn_lyr1[h] = activated;
21     wj += 1; bi += 1;
22   }
23
24   wj = 0; bi = 0;
25   for(int i = 0; i < 10; i++){
26     tmp = 0.0; wi = 0;
27     for(int r = 0; r < size_hdn_layer; r++){
28       tmp += (hdn_lyr1[r] * w2[wi][wj]);
29       wi += 1;
30     }
31     activated = actFunc(tmp + b2[bi]);
32     output[i] = activated;
33     wj += 1; bi += 1;
34   }
35   predict(output, 10);
36 }
```

Kode Sumber 4.32 Program Fungsi Model Jaringan Saraf Tiruan

4.5.10. Implementasi Fungsi Aktivasi

Subbab ini menjabarkan fungsi aktivasi yang digunakan untuk menormalkan hasil perkalian input dan bobot pada fungsi JST. Fungsi aktivasi yang digunakan pada tugas akhir ini adalah fungsi aktivasi Sigmoid. Implementasi tahap ini dapat dilihat pada kode sumber 4.33.

```
1 double actFunc(double x){
2   return (1/(1 + exp(-x)));
3 }
```

Kode Sumber 4.33 Program Fungsi Aktivasi

4.5.11. Implementasi Fungsi Prediksi Kelas Digit

Subbab ini menjabarkan fungsi untuk menghitung prediksi kelas (0-9) dari digit. Implementasi tahap ini dapat dilihat pada kode sumber 4.34.

```
 1 void predict(double output[], int size){
 2   int dgt_pred;
 3   double tmp = 0.0;
 4
 5   for(int i = 0; i < size; i++){
 6     if(output[i] > tmp){
 7       tmp = output[i];
 8       dgt_pred = i;
 9     }
10   }
11   ans.push_back(dgt_pred);
12 }
```

Kode Sumber 4.34 Program Fungsi Prediksi Kelas Digit

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada tugas akhir ini.

5.1. Lingkungan Uji Coba

Lingkungan uji coba digunakan untuk uji coba kebenaran adalah salah satu sistem yang digunakan oleh situs penilaian daring SPOJ pada permasalahan *Hard Image Recognition* (HIR) dengan spesifikasi sebagai berikut:

1. Perangkat Keras
   (a) Processor Intel Xeon E3-1220 v5 (5 CPUs)
   (b) Random Access Memory 1536MB
2. Perangkat Lunak
   (a) Kompiler GCC 6.3.0

Lingkungan uji coba yang digunakan untuk melakukan uji coba kinerja menggunakan komputer pribadi penulis.

5.1.1. Program Pembuatan Dataset, Pelatihan Fungsi JST, dan Optimasi GA

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700 CPU @ 3.4GHz (4 CPUs), 3.4GHz
   (b) Random Access Memory 24576MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Jupyter Notebook

(c) Bahasa Pemrograman Python 3.6
(d) MinGW 64-bit

5.1.2. Program Pengujian Fungsi JST

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700 CPU @ 3.4GHz (4 CPUs), 3.4GHz
   (b) Random Access Memory 24576MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) MinGW 64-bit

5.2. Skenario Uji Coba

Subbab ini akan menjelaskan hasil pengujian program untuk menyelesaiakan permasalahan Hard Image Recognition. Pengujian dilakukan untuk perbandingan metode penyelesaian dengan menggunakan algoritma Jaringan Saraf Tiruan tanpa dilakukan optimasi dan metode penyelesaian dengan menggunakan algoritma Jaringan Saraf Tiruan dengan optimasi menggunakan *Genetic Algorithm*. Metode pengujian yang dilakukan adalah sebagai berikut.

1. Pengujian luar. Pengujian ini menggunakan Online Judge untuk menguji kebenaran dan kinerja program.
2. Pengujian lokal. Pengujian ini menggunakan mesin yang digunakan dalam pengembangan untuk mengukur kinerja program.

Dalam pengujian lokal, penulis menggunakan dataset yang didapat dari situs diskusi *online* pada SPOJ pada permasalahan Hard Image Recognition[2] dengan menggunakan banyak *neuron* yang

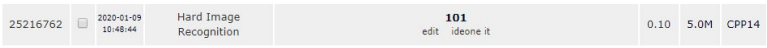
didapat dari penelitian sebelumnya dibandingkan dengan menggunakan banyak *neuron* yang telah dioptimasi menggunakan *Genetic Algorithm*. Penulis juga melakukan pengujian secara lokal untuk digit yang mengalami derau coretan. Selain itu untuk pengujian lokal, penulis menggunakan dataset yang telah dibangun dan mengukur kinerja dari *Genetic Algorithm*.

Untuk pengujian kebenaran, uji coba dilakukan dengan mengirimkan kode program dengan bobot yang didapat dari pelatihan JST dengan menggunakan banyak *neuron* yang didapat dari penelitian sebelumnya dibandingkan dengan menggunakan *neuron* yang telah dioptimasi menggunakan *Genetic Algorithm*. Penulis juga melakukan perbandingan untuk program dengan praproses penghilangan derau coretan dengan tanpa melakukan praproses penghilangan derau coretan. Selain itu untuk pengujian luar, uji coba dilakukan dengan mengirimkan kode program dengan algoritma JST, algoritma JST dengan optimasi GA, dan kode program dengan praproses penghilangan derau coretan ke situs penilaian *online* SPOJ sebanyak 10 kali.

5.3. Uji Coba Kebenaran

Pada subbab ini akan dibahas mengenai uji coba kebenaran yang dilakukan dengan mengirim kode sumber terkait ke dalam situs penilaian daring SPOJ. Berikut bukti hasil pengujian.

1. Kode sumber dengan banyak *neuron* 40 (Gambar 5.1)

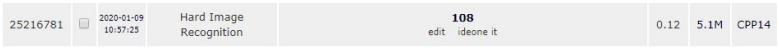


Gambar 5.1 Umpan Balik JST dengan *Neuron* sebanyak 40

2. Kode sumber dengan banyak *neuron* hasil optimasi GA, yaitu 60 (Gambar 5.2)

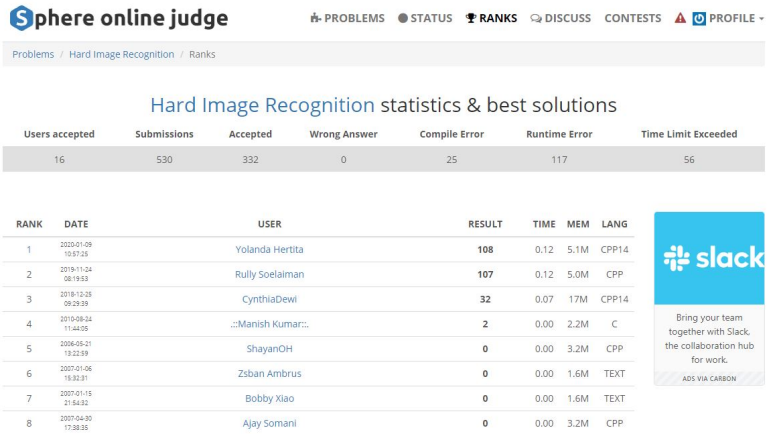
Gambar 5.2 Umpan Balik JST dengan *Neuron* hasil optimasi GA

3. Kode sumber dengan banyak *neuron* hasil optimasi GA, yaitu 60 dan praproses penghilangan derau coretan (Gambar 5.3)



Gambar 5.3 Umpan Balik JST dengan kode praproses penghilangan derau coretan

Selain itu, penulis berhasil mendapatkan peringkat pertama dengan nilai tertinggi pada permasalahan SPOJ Hard Image Recognition. Bukti peringkat pertama dapat dilihat pada Gambar 5.4.



Gambar 5.4 Peringkat Pada Permasalahan SPOJ Hard Image Recognition

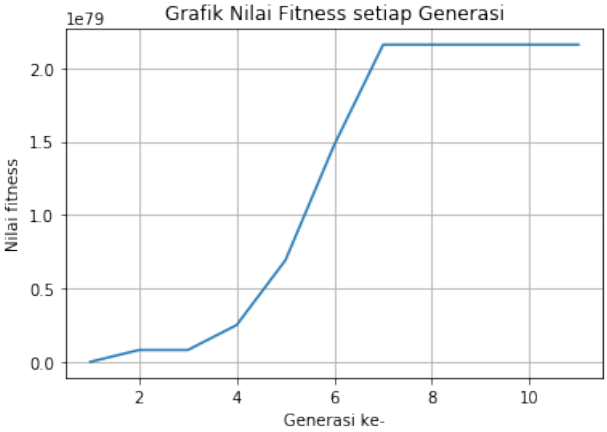
5.4. Uji Coba Kinerja Lokal

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari *Genetic Algorithm* untuk mengoptimasi banyak *neuron* dari Jaringan Saraf Tiruan. Pengujian kinerja dilakukan dengan melakukan komputasi sebanyak maksimum 300 generasi, dengan batas maksimum 5 generasi berturut-turut yang tidak memiliki perubahan pada nilai *fitness*.

```
Generation:  6
Best n neurons for hidden layer:  61
Best fitness:  1.4790132430525831e+79
Generation:  7
Best n neurons for hidden layer:  60
Best fitness:  2.159643389591646e+79
Generation:  8
Best n neurons for hidden layer:  60
Best fitness:  2.159643389591646e+79
Generation:  9
Best n neurons for hidden layer:  60
Best fitness:  2.159643389591646e+79
Generation:  10
Best n neurons for hidden layer:  60
Best fitness:  2.159643389591646e+79
Generation:  11
Best n neurons for hidden layer:  60
Best fitness:  2.159643389591646e+79
Wall time: 5min 36s
```

Gambar 5.5 Contoh Luaran pada Setiap Generasi

Pada Gambar 5.5, dapat dilihat generasi terhenti pada generasi ke-11 dengan total waktu eksekusi 5 menit 36 detik. Banyak *neuron* optimal dari optimasi GA adalah sebanyak 60 *neuron*. Grafik nilai *fitness* dapat dilihat pada Gambar 5.6 dan tabel luaran setiap generasi dapat dilihat pada Tabel 5.1.



Gambar 5.6 Grafik Nilai Fitness setiap Generasi

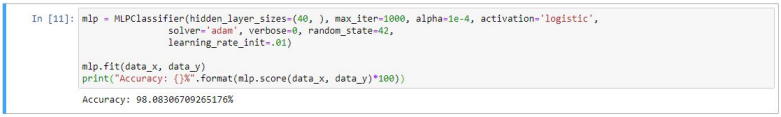
Tabel 5.1 Banyak Neuron Optimal dan Nilai Fitness Terbaik pada Setiap Generasi

Generasi	Banyak Neuron Optimal	Nilai Fitness Terbaik
1	41	1.112198e+76
2	43	8.184597e+77
3	43	8.184597e+77
4	46	2.517529e+78
5	62	6.936687e+78
6	61	1.479013e+79
7	60	2.159643e+79
8	60	2.159643e+79
9	60	2.159643e+79
10	60	2.159643e+79
11	60	2.159643e+79

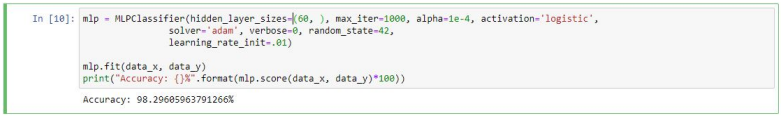
5.5. Evaluasi Kebenaran Uji Coba Lokal

Evaluasi dilakukan dengan memeriksa hasil keluaran program apakah sama dengan keluaran yang diharapkan dari SPOJ permasalahan Hard Image Recognition. Pada bagian ini akan ditampilkan evaluasi hasil dari algoritma JST tanpa dilakukan optimasi GA dan algoritma JST dengan optimasi GA.

Pengujian dilakukan terhadap data *training* menggunakan dataset yang telah dibangun sebelumnya. Hasil pengujian dengan menggunakan algoritma JST tanpa dilakukan optimasi GA mendapatkan akurasi pada *training* sebesar 98.083%. Hasil pengujian dengan menggunakan algoritma JST dengan optimasi GA mendapatkan akurasi pada *training* sebesar 98.296%. Bukti pengujian ini terdapat pada Gambar 5.7 dan 5.8.



Gambar 5.7 Hasil Pengujian Algoritma JST pada Data Training



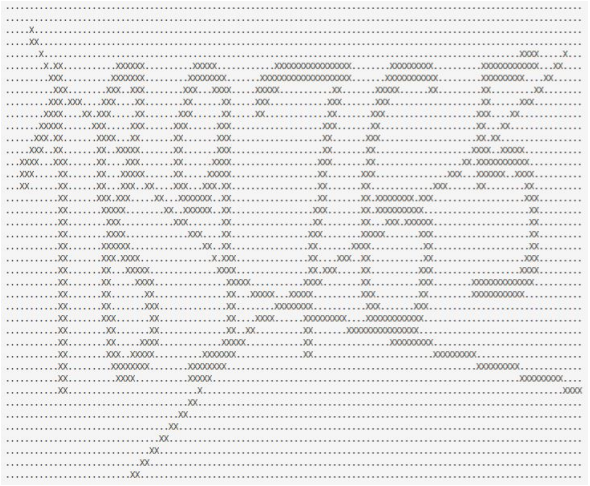
Gambar 5.8 Hasil Pengujian Algoritma JST dengan Optimasi GA pada Data Training

Pengujian juga dilakukan terhadap dataset yang didapat dari situs diskusi online SPOJ pada permasalahan Hard Image Recognition. Pada Tabel 5.2 merupakan perbandingan kebenaran dari algoritma JST tanpa dilakukan optimasi GA dan algoritma JST dengan optimasi GA. Detil masukan dan luaran dari program dapat dilihat pada Lampiran.

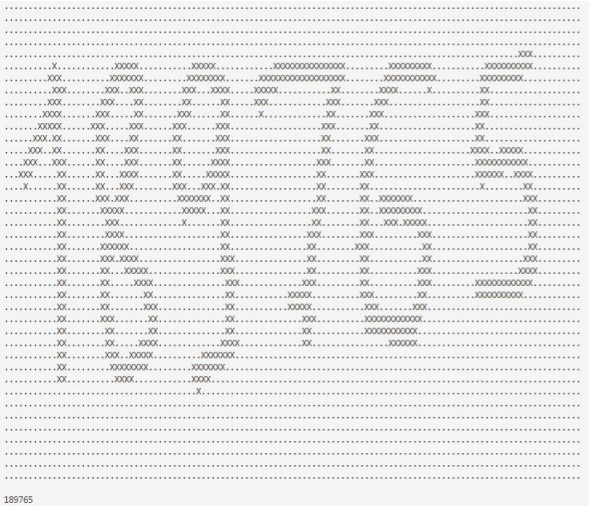
Tabel 5.2 Perbandingan hasil prediksi JST tanpa GA dan prediksi JST dengan GA

Masukan	Prediksi JST tanpa GA	Prediksi JST dengan GA
177188	177188	177188
388910	388910	388910
381729	381729	381729
254839	254839	254839
349010	349010	349010
491881	491881	491881
457840	457840	457840
189765	198848	189765
371981	371981	371981
271891	271891	271891
000482	000882	000832
567843	567842	567842
239001	239001	239001

Program dengan praproses penghilangan derau coretan dapat mengenali citra yang mengalami coretan. Pada Gambar 5.10 merupakan hasil dari proses *inpainting* citra awal 5.9 yang merupakan salah satu contoh citra yang mengalami derau coretan.



Gambar 5.9 Citra yang Mengalami Derau Coretan



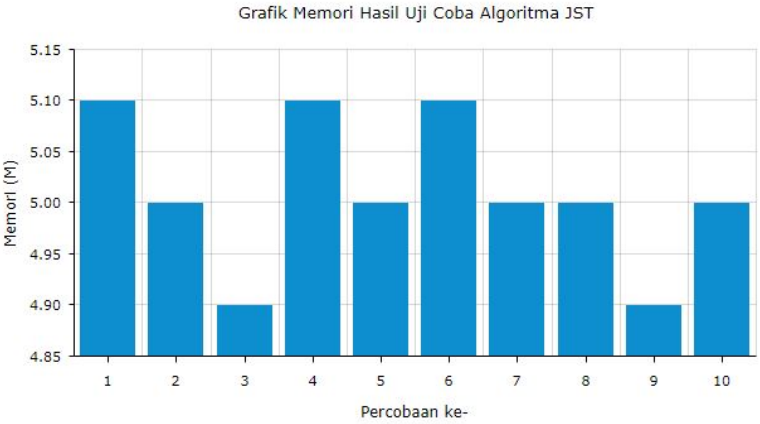
Gambar 5.10 Hasil Penhilangan Derau Coretan pada Citra 5.9

5.6. Uji Coba Kinerja Luar

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari algoritma JST dengan menggunakan banyak *neuron* tanpa menggunakan optimasi GA dibandingkan dengan menggunakan banyak *neuron* yang telah dioptimasi menggunakan GA. Pengujian dilakukan dengan cara mengirimkan kode program ke situs penilaian *online* SPOJ.

Rata-rata hasil pengumpulan kode berkas dengan algoritma JST dengan menggunakan banyak *neuron* tanpa menggunakan optimasi GA adalah 0.109 detik dengan memori 5.01 MB. Grafik uji coba algoritma ini dapat dilihat pada Gambar 5.11 dan 5.12. Rata-rata hasil pengumpulan kode berkas dengan algoritma JST dengan menggunakan banyak *neuron* yang telah dioptimasi menggunakan GA adalah 0.130 detik dengan memori 5.14 MB. Grafik uji coba algoritma ini dapat dilihat pada Gambar 5.13 dan 5.14. Rata-rata hasil pengumpulan kode berkas algoritma dengan praproses penghilangan derau coretan adalah 0.129 detik dengan memori 5.10 MB. Grafik uji coba algoritma ini dapat dilihat pada Gambar 5.15 dan 5.16. Detil mengenai hasil uji kinerja dapat dilihat pada Gambar 5.17, 5.18, dan 5.19.

Gambar 5.11 Grafik Waktu Hasil Uji Coba Algoritma JST pada Situs SPOJ

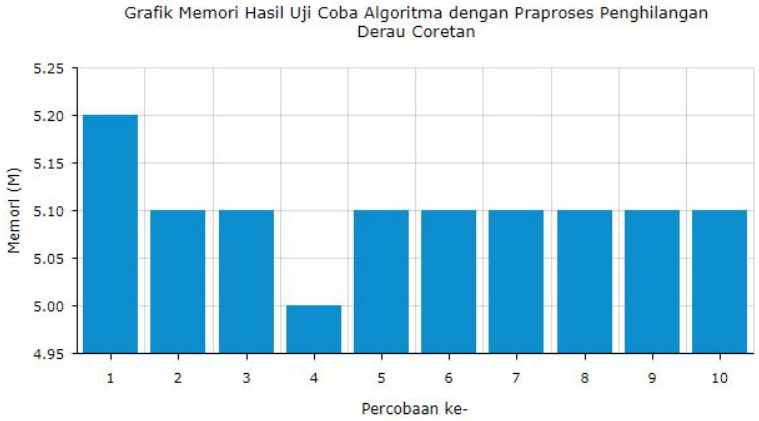


Gambar 5.12 Grafik Memori Hasil Uji Coba Algoritma JST pada Situs SPOJ

Gambar 5.13 Grafik Waktu Hasil Uji Coba Algoritma JST dengan Optimasi GA pada Situs SPOJ

Gambar 5.14 Grafik Memori Hasil Uji Coba Algoritma JST dengan Optimasi GA pada Situs SPOJ

Gambar 5.15 Grafik Waktu Hasil Uji Coba Algoritma dengan Praproses Penghilangan Derau Coretan pada Situs SPOJ



Gambar 5.16 Grafik Memori Hasil Uji Coba Algoritma dengan Praproses Penghilangan Derau Coretan pada Situs SPOJ

ID	DATE	PROBLEM	RESULT	TIME	MEM	LANG
25217074	2020-01-09 12:17:10	Hard Image Recognition	**101** edit ideone it	0.10	5.1M	CPP14
25217073	2020-01-09 12:17:06	Hard Image Recognition	**101** edit ideone it	0.10	5.0M	CPP14
25217069	2020-01-09 12:16:00	Hard Image Recognition	**101** edit ideone it	0.10	4.9M	CPP14
25217068	2020-01-09 12:15:54	Hard Image Recognition	**101** edit ideone it	0.10	5.1M	CPP14
25217066	2020-01-09 12:15:48	Hard Image Recognition	**101** edit ideone it	0.11	5.0M	CPP14
25217065	2020-01-09 12:15:44	Hard Image Recognition	**101** edit ideone it	0.11	5.1M	CPP14
25217064	2020-01-09 12:15:38	Hard Image Recognition	**101** edit ideone it	0.12	5.0M	CPP14
25217063	2020-01-09 12:15:33	Hard Image Recognition	**101** edit ideone it	0.10	5.0M	CPP14
25217062	2020-01-09 12:15:27	Hard Image Recognition	**101** edit ideone it	0.14	4.9M	CPP14
25217060	2020-01-09 12:15:14	Hard Image Recognition	**101** edit ideone it	0.11	5.0M	CPP14

Gambar 5.17 Hasil Pengumpulan Kode Program Utama Algoritma JST

ID	DATE	PROBLEM	RESULT	TIME	MEM	LANG
25217115	2020-01-09 12:25:06	Hard Image Recognition	**106** edit ideone it	0.13	5.2M	CPP14
25217113	2020-01-09 12:25:00	Hard Image Recognition	**106** edit ideone it	0.13	5.2M	CPP14
25217112	2020-01-09 12:24:53	Hard Image Recognition	**106** edit ideone it	0.13	5.2M	CPP14
25217111	2020-01-09 12:24:51	Hard Image Recognition	**106** edit ideone it	0.13	5.1M	CPP14
25217102	2020-01-09 12:22:04	Hard Image Recognition	**106** edit ideone it	0.13	5.2M	CPP14
25217100	2020-01-09 12:21:55	Hard Image Recognition	**106** edit ideone it	0.13	5.1M	CPP14
25217099	2020-01-09 12:21:48	Hard Image Recognition	**106** edit ideone it	0.13	5.0M	CPP14
25217092	2020-01-09 12:20:41	Hard Image Recognition	**106** edit ideone it	0.13	5.1M	CPP14
25217091	2020-01-09 12:20:34	Hard Image Recognition	**106** edit ideone it	0.13	5.1M	CPP14
25217090	2020-01-09 12:20:27	Hard Image Recognition	**106** edit ideone it	0.13	5.2M	CPP14

Gambar 5.18 Hasil Pengumpulan Kode Program Utama Algoritma JST dengan Optimasi GA

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
25217035	☐	2020-01-09 12:10:55	Hard Image Recognition	**108** edit ideone it	0.13	5.1M	CPP14
25217032	☐	2020-01-09 12:10:08	Hard Image Recognition	**108** edit ideone it	0.14	5.1M	CPP14
25217030	☐	2020-01-09 12:09:50	Hard Image Recognition	**108** edit ideone it	0.12	5.1M	CPP14
25217027	☐	2020-01-09 12:09:31	Hard Image Recognition	**108** edit ideone it	0.12	5.0M	CPP14
25217022	☐	2020-01-09 12:08:33	Hard Image Recognition	**108** edit ideone it	0.12	5.1M	CPP14
25216888	☐	2020-01-09 11:29:35	Hard Image Recognition	**108** edit ideone it	0.12	5.1M	CPP14
25216878	☐	2020-01-09 11:28:31	Hard Image Recognition	**108** edit ideone it	0.15	5.1M	CPP14
25216875	☐	2020-01-09 11:28:18	Hard Image Recognition	**108** edit ideone it	0.13	5.1M	CPP14
25216846	☐	2020-01-09 11:23:38	Hard Image Recognition	**108** edit ideone it	0.14	5.1M	CPP14
25216845	☐	2020-01-09 11:22:35	Hard Image Recognition	**108** edit ideone it	0.12	5.1M	CPP14

Gambar 5.19 Hasil Pengumpulan Kode Program Utama Algoritma dengan Praproses Penghilangan Derau Coretan

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Berdasarkan penjabaran di bab-bab sebelumnya, dapat disimpulkan beberapa poin terkait penyelesaian permasalahan Hard Image Recognition.

1. Sistem yang diuji secara lokal berhasil mengambil elemen digit pada citra dan memprediksi citra yang mengalami derau coretan dengan benar.
2. Penggunaan berbagai macam jenis font pada pembuatan dataset dapat meningkatkan akurasi pada permasalahan SPOJ Hard Image Recognition.
3. Algoritma Jaringan Saraf Tiruan yang diuji secara lokal berhasil memprediksi 11 dari 13 contoh citra masukan dari dataset pada diskusi SPOJ Hard Image Recognition[2] dengan benar dan berhasil mendapat akurasi 98.296% dalam pengujian terhadap dataset citra pelatihan.
4. Arsitektur banyak *neuron* dari Jaringan Saraf Tiruan dapat dioptimasi menggunakan *Genetic Algorithm* sehingga tidak perlu dilakukan percobaan setiap *neuron* dari 1 sampai dengan N secara manual.
5. Sistem yang diajukan berhasil memprediksi 108 citra masukan dengan benar pada situs penilaian daring SPOJ Hard Image Recognition, lebih tinggi apabila dibandingkan metode JST yang dilakukan tanpa menggunakan optimasi dan tanpa peng-

hilangan derau coretan.
6. Sistem yang didapatkan nilai tertinggi menggunakan *hidden layer* sebanyak 1 dan *neuron* sebanyak 60.

6.2. Saran

Pada Tugas Akhir kali ini tentunya terdapat kekurangan serta nilai-nilai yang dapat penulis ambil. Berikut adalah saran-saran yang dapat digunakan untuk pengembangan di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan uji coba yang telah dilakukan.

1. Walaupun nilai yang didapat merupakan nilai tertinggi pada sistem penilaian daring SPOJ Hard Image Recognition, tidak menutup kemungkinan penerapan metode atau perspektif lain dapat mendapatkan nilai yang lebih tinggi.
2. Usaha meningkatkan nilai yang didapat pada sistem penilaian daring SPOJ Hard Image Recognitiondapat dilakukan dengan menentukan dataset yang optimal seperti melakukan proses augmentasi untuk penambahan variasi data.
3. Penambahan atau pemilihan jenis *font* yang tepat dapat membantu untuk meningkatkan akurasi.
4. Melakukan tahap praproses yang lebih baik untuk meningkatkan nilai.

DAFTAR PUSTAKA

[1] SPOJ. (2015). Hard Image Recognition, [Online]. Available: https://www.spoj.com/problems/HIR/.

[2] D. SPOJ. (2006). Hard Image Recognition Open Contest 2006, [Online]. Available: http://discuss.spoj.com/t/hard-image-recognition/663.

[3] A. Brink, H. van der Klauw, and L. Schomaker, "Automatic removal of crossed-out handwritten text and the effect on writer verification and identification", 68150A, 2008. doi: 10.1117/12.766466. [Online]. Available: https://doi.org/10.1117/12.766466.

[4] Anonim. (2009). Nearest Neighbor Image Scaling, [Online]. Available: http://tech-algorithm.com/articles/nearest-neighbor-image-scaling/.

[5] C. D. Tejakusuma, R. Soelaiman, and W. N. Khotimah, *Tugas Akhir IF184802 - Implementasi Jaringan Saraf Tiruan Untuk Pengenalan Karakter Pada Studi Kasus Permasalahan SPOJ Hard Image Recognition*. 2019.

[6] D. Authors. (2000). Dafont, [Online]. Available: https://www.dafont.com/.

[7] S. Haykin, *Neural Networks and Learning Machines*, third edition. 2009.

[8] D. Gupta. (2017). Fundamentals of Deep Learning – Activation Functions and When to Use Them?, [Online]. Available: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[9] Wikipedia. (2017). Stochastic Gradient Descent, [Online]. Available: https : / / en . wikipedia . org / wiki / Stochastic \ _gradient_descent.

[10] B. P. G and V. G. C, "Optimizing feedforward artificial neural network architecture", *Engineering Applications of Artificial Intelligence*, vol. 20, no. 365-382, 2007. [Online]. Available: https://doi.org/10.1016/j.engappai.2006.06.005.

[11] B. Coppin, *Artificial Intelligence Illuminated*, first edition. 2004.

[12] TutorialsPoint. (2016). Genetic Algorithms - Parent Selection, [Online]. Available: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm.

[13] Wikipedia. (2019). Stochastic Universal Sampling, [Online]. Available: https : / / en . wikipedia . org / wiki / Stochastic \ _universal_sampling.

[14] ——, (2019). Tipografi, [Online]. Available: https : / / id . wikipedia.org/wiki/Tipografi.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, third edition. 2009.

[16] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*, first edition. 2006.

LAMPIRAN A: Bobot dan Bias pada Program Pengujian C++

Bobot pada *hidden layer* 1 { {0.32, -0.16, 0.32, 0.27, -0.11, 0.18, 0.48, 0.03, -0.13, 0.15, 0.01, 0.13, -0.05, 0.07, 0.24, -0.21, -0.46, -0.15, 0.29, 0.25, 0.01, 0.05, 0.67, 0.25, -0.46, -0.26, -0.31, 0.09, 0.49, -0.03, -0.48, 0.05, -0.05, -0.18, -0.27, 0.1, 0.3, -0.55, -0.81, 0.24, 0.32, -0.51, -0.15, -0.25, 0.47, -0.55, -0.01, -0.6, 0.11, -0.12, 0.28, -0.16, 0.33, 0.5, -0.24, 0.01, -0.38, -0.01, -0.21, 0.11 }, {0.32, -0.26, 0.33, 0.23, -0.1, 0.23, 0.49, 0.03, -0.2, 0.18, 0.11, 0.03, -0.15, 0.15, 0.31, -0.13, -0.4, -0.21, 0.28, 0.22, 0.04, 0.12, 0.67, 0.21, -0.48, -0.32, -0.24, 0.11, 0.53, 0.03, -0.55, 0.12, 0.04, -0.23, -0.3, 0.06, 0.33, -0.51, -0.9, 0.2, 0.31, -0.49, -0.11, -0.31, 0.55, -0.61, 0.01, -0.57, 0.07, -0.13, 0.19, -0.24, 0.33, 0.48, -0.24, 0.01, -0.28, -0.01, -0.09, 0.14 }, {0.39, 0.02, 0.02, 0.32, -0.35, 0.41, 0.96, -0.63, -0.2, 0.13, 0.64, 0.05, -0.05, -0.04, 0.79, 0.01, -0.86, -0.07, 0.44, -0.61, -0.09, 0.31, 0.07, 0.75, -0.52, -0.54, -0.52, 0.22, 0.22, -0.47, 0.16, 0.84, 0.06, 0.63, -0.27, 0.07, 0.36, -0.73, -1.17, -0.15, -0.43, -0.38, -0.11, -0.23, 0.49, -0.3, -0.18, -0.42, 0.12, 0.03, 0.4, 0.43, 0.7, 0.79, -0.26, -0.27, 0.22, -0.28, -0.28, 0.42 }, {-0.14, 0.14, 0.11, 0.39, -0.06, 0.6, 0.63, -0.66, -0.07, 0.07, 0.5, 0.12, 0.09, 0.03, 0.66, 0.11, -0.84, -0.08, 0.28, -0.56, -0.16, 0.08, -0.25, 0.74, -0.15, -0.08, -0.39, 0.38, 0.29, -0.56, -0.02, 0.68, 0.13, 0.38, -0.19, 0.19, 0.07, -0.22, -0.64, -0.26, -0.18, -0.24, -0.35, -0.59, 0.14, -0.21, -0.06, -0.25, 0.08, 0.07, 0.43, 0.12, 0.42, 0.77, 0.01, -0.13, 0.36, -0.18, -0.46, 0.01 }, {0.04, 0.13, -0.0, -0.28, -0.16, 0.49, 0.39, -0.33, 0.04, 0.11, 0.12, 0.05, 0.07, -0.68, 0.49, -0.15, -0.62, -0.07, 0.01, -0.2, -0.22, 0.18, -0.09, 0.53, -0.23, 0.22, -0.28, 0.08, 0.02, -0.24, -0.1, 0.25, -0.01, 0.25, -0.26, -0.15, 0.03, 0.09, -0.2, -0.31, -0.11, 0.03, -0.4, 0.12, 0.03, -0.17, -0.01, 0.22, -0.13, 0.03, 0.28, 0.11, 0.11, 0.34, -0.11, -0.03, -0.09, -0.1, 0.2, -0.05 }, {-0.12, 0.09, 0.09, 0.14, -0.09, 0.22, 0.35, -0.36, 0.09, 0.04, 0.15, 0.01, -0.03, 0.11, 0.31, 0.29, -0.23, 0.25, -0.22, 0.22, 0.16, 0.06, 0.08, 0.09, -0.48, -

0.06, 0.14, -0.17, -0.13, -0.02, 0.13, 0.22, 0.16, 0.25, -0.29, 0.04, 0.1, -0.15, -0.16, -0.44, -0.07, 0.11, 0.06, 0.32, 0.04, 0.06, -0.08, 0.01, -0.41, 0.06, 0.45, 0.1, -0.07, 0.31, -0.06, 0.02, -0.13, -0.29, 0.49, -0.03 }, {-0.02, 0.6, 0.01, -0.01, -0.04, -0.2, 0.4, -0.07, 0.02, 0.02, 0.11, 0.01, -0.02, 0.16, 0.1, 0.05, 0.11, 0.1, 0.34, 0.21, -0.06, 0.19, -0.1, 0.24, -0.12, 0.05, -0.04, -0.37, 0.17, 0.23, 0.43, -0.03, 0.11, 0.15, -0.09, -0.02, 0.04, -0.07, -0.02, -0.32, 0.02, 0.26, 0.43, 0.5, -0.02, 0.12, -0.17, -0.05, 0.2, -0.01, 0.04, -0.12, -0.18, 0.12, -0.07, 0.31, 0.15, 0.1, 0.38, 0.4 }, {0.18, 0.21, 0.07, 0.1, 0.11, -0.12, -0.23, -0.07, 0.12, 0.02, 0.14, 0.11, 0.07, 0.2, -0.19, -0.14, -0.04, -0.11, 0.33, -0.28, 0.04, 0.44, 0.18, 0.04, -0.34, 0.35, -0.09, -0.02, -0.04, 0.1, 0.57, -0.16, 0.04, 0.17, -0.34, -0.16, -0.04, 0.3, 0.33, 0.15, 0.03, 0.47, 0.69, 0.63, -0.14, 0.2, -0.24, 0.16, 0.36, 0.12, 0.08, 0.17, 0.02, -0.05, 0.17, 0.28, -0.09, 0.01, 0.37, 0.18 }, {-0.11, 0.14, -0.02, 0.05, 0.18, -0.45, -0.3, -0.05, 0.16, 0.12, 0.06, 0.09, 0.01, 0.06, -0.04, 0.16, -0.13, 0.0, 0.18, 0.43, -0.29, -0.15, 0.17, -0.11, 0.28, 0.22, 0.23, 0.31, 0.1, 0.39, 0.4, -0.01, 0.06, 0.27, -0.01, 0.25, 0.01, 0.2, 0.32, -0.09, 0.02, 0.15, 0.34, 0.43, -0.11, 0.17, 0.02, 0.16, 0.2, 0.09, -0.15, -0.0, 0.02, -0.56, -0.07, 0.45, -0.01, 0.39, 0.48, -0.07 }, {0.07, 0.13, -0.08, 0.08, 0.01, -0.43, -0.25, -0.14, 0.1, 0.09, 0.17, 0.11, 0.04, 0.12, 0.03, 0.28, -0.19, 0.14, 0.03, 0.13, -0.26, -0.17, 0.09, -0.26, 0.07, 0.1, 0.06, 0.18, -0.03, 0.25, 0.55, 0.13, 0.01, 0.28, 0.11, 0.26, -0.04, 0.16, 0.3, -0.07, 0.13, 0.3, 0.43, 0.43, -0.09, 0.19, 0.02, -0.08, -0.1, -0.02, -0.04, -0.25, -0.09, -0.4, -0.09, 0.34, 0.19, 0.22, 0.62, 0.03 }, {0.33, -0.12, -0.04, 0.16, -0.29, -0.4, -0.08, -0.27, 0.04, 0.08, 0.28, 0.11, 0.03, 0.31, 0.15, 0.56, -0.08, 0.2, 0.14, -0.28, -0.19, -0.2, 0.21, -0.29, 0.04, 0.24, 0.07, 0.08, -0.37, 0.0, 0.06, 0.08, -0.06, 0.55, -0.0, -0.19, 0.0, 0.23, 0.25, -0.05, -0.24, 0.24, 0.59, 0.09, -0.19, 0.25, 0.09, 0.06, 0.18, -0.0, -0.26, 0.07, -0.18, -0.45, -0.24, 0.38, 0.12, 0.0, 0.47, 0.07 }, {0.44, 0.21, -0.12, 0.05, -0.46, -0.28, 0.17, -0.16, -0.11, 0.03, 0.47, 0.12, -0.05, 0.03, 0.02, -0.01, -0.13, 0.31, -0.02, 0.08, -0.19, 0.17, 0.23, 0.26, 0.09, -0.07, -0.14, 0.0, -0.07, 0.07, 0.1, 0.38, -0.22, 0.3, -0.16, 0.03, 0.0, -0.33, -0.08, -0.18,

-0.26, 0.15, 0.31, 0.19, -0.08, 0.21, -0.08, -0.02, 0.2, 0.02, 0.03, 0.26, 0.03, 0.14, -0.14, 0.07, 0.01, -0.03, 0.07, 0.28 }, {0.44, -0.5, 0.12, 0.24, -0.45, 0.58, 0.29, -0.26, -0.06, 0.01, -0.32, 0.11, 0.24, -0.29, -0.22, 0.05, -0.16, -0.2, -0.22, -0.49, 0.2, 0.1, 0.28, 0.16, -0.45, 0.08, -0.15, 0.07, -0.46, -0.16, -0.03, -0.14, -0.25, 0.49, -0.12, -0.21, 0.14, 0.1, -0.25, 0.32, -0.4, 0.19, 0.22, 0.26, -0.4, 0.15, 0.18, -0.14, -0.22, 0.23, -0.14, 0.42, 0.01, 0.08, 0.23, -0.08, 0.01, -0.34, 0.07, 0.22 }, {0.47, -0.32, 0.32, 0.06, -0.56, 0.64, 0.45, -0.39, -0.06, 0.12, -0.32, 0.08, 0.24, -0.31, -0.12, -0.12, -0.22, -0.2, -0.34, -0.41, 0.06, 0.43, 0.34, 0.58, -0.05, -0.11, -0.15, 0.2, -0.27, -0.23, -0.26, -0.1, -0.31, 0.45, -0.18, -0.05, 0.2, -0.15, -0.34, 0.18, -0.27, 0.21, 0.33, -0.03, -0.25, 0.12, 0.2, -0.06, -0.04, 0.21, 0.01, 0.58, 0.11, 0.52, 0.4, -0.14, -0.41, -0.09, -0.02, 0.28 }, {0.68, -0.17, 0.11, 0.1, -0.42, 0.74, 0.69, -0.02, -0.08, -0.01, -0.61, 0.1, 0.26, -0.26, 0.22, -0.11, -0.53, -0.47, 0.42, -0.67, -0.4, 0.51, 0.32, 0.62, -0.61, -0.22, -0.06, 0.37, -0.17, -0.32, -0.58, -0.07, 0.06, 0.46, -0.24, -0.3, 0.14, -0.49, -0.66, -0.2, -0.32, -0.04, -0.25, -0.36, -0.21, -0.07, 0.26, -0.13, 0.45, 0.2, 0.3, 0.76, 0.28, 0.21, 0.13, 0.11, -0.17, 0.27, -0.23, 0.58 }, {0.49, -0.72, 0.29, 0.04, -0.59, 0.95, 0.73, -0.62, -0.06, -0.05, -0.78, 0.05, 0.37, -0.39, 0.09, 0.13, -0.84, -0.44, 0.08, -0.95, -0.62, 0.53, 1.07, 0.64, -0.61, -0.06, -0.42, 0.46, -0.41, -0.84, -0.06, -0.06, 0.23, 0.85, -0.21, -0.25, 0.09, -0.17, -0.38, -0.01, 0.02, -0.06, 0.12, 0.09, -0.37, -0.2, 0.28, -0.14, 0.17, 0.29, 0.03, 0.91, 0.26, -0.2, 0.1, -0.0, 0.12, -0.0, 0.23, 0.56 }, {0.28, -0.25, 0.27, 0.24, -0.09, 0.28, 0.53, -0.04, -0.13, 0.12, 0.03, 0.12, -0.06, 0.07, 0.22, -0.2, -0.41, -0.26, 0.3, 0.29, -0.01, 0.13, 0.64, 0.21, -0.42, -0.3, -0.24, 0.07, 0.43, 0.08, -0.48, 0.06, -0.02, -0.23, -0.35, -0.03, 0.38, -0.54, -0.77, 0.28, 0.35, -0.5, -0.04, -0.29, 0.5, -0.54, 0.05, -0.55, 0.07, -0.08, 0.17, -0.14, 0.27, 0.38, -0.22, 0.01, -0.36, 0.04, -0.1, 0.07 }, {0.36, -0.29, 0.35, 0.31, -0.0, 0.3, 0.55, 0.03, -0.16, 0.16, -0.05, 0.08, -0.12, 0.06, 0.23, -0.14, -0.45, -0.17, 0.24, 0.26, 0.01, 0.14, 0.67, 0.28, -0.4, -0.28, -0.3, 0.02, 0.51, -0.02, -0.47, 0.03, 0.06, -0.2, -0.27, 0.01, 0.36, -0.51, -0.84, 0.31, 0.44, -0.44, -0.05, -0.33, 0.45, -0.53, -0.01, -0.65,

0.08, -0.11, 0.18, -0.24, 0.15, 0.38, -0.18, -0.05, -0.34, 0.06, -0.19, 0.19 }, {0.39, -0.12, 0.03, 0.41, -0.26, 0.48, 0.9, -0.72, -0.07, 0.1, 0.66, 0.1, 0.02, -0.01, 0.68, -0.0, -0.9, -0.18, 0.49, -0.69, -0.19, 0.4, 0.1, 0.82, -0.42, -0.52, -0.55, 0.24, 0.3, -0.47, 0.13, 0.78, 0.07, 0.63, -0.31, 0.06, 0.28, -0.71, -1.16, -0.18, -0.34, -0.33, -0.08, -0.21, 0.53, -0.33, -0.22, -0.37, 0.16, -0.01, 0.38, 0.48, 0.59, 0.79, -0.25, -0.29, 0.25, -0.26, -0.35, 0.48 }, {-0.05, 0.12, 0.17, 0.41, -0.18, 0.61, 0.57, -0.64, -0.01, 0.15, 0.53, 0.11, 0.07, 0.16, 0.6, 0.3, -0.81, -0.1, 0.26, -0.67, -0.01, 0.11, -0.33, 0.71, -0.25, 0.03, -0.24, 0.32, 0.15, -0.59, -0.28, 0.63, 0.13, 0.49, -0.27, 0.06, 0.15, -0.24, -0.64, -0.31, -0.35, -0.23, -0.23, -0.59, 0.06, -0.09, -0.17, -0.27, -0.11, 0.09, 0.3, 0.18, 0.23, 0.78, -0.01, -0.06, 0.58, -0.25, -0.44, -0.03 }, {0.14, 0.04, -0.02, 0.01, -0.08, 0.4, 0.33, -0.34, -0.0, 0.13, 0.09, 0.06, -0.02, -0.3, 0.42, 0.04, -0.44, -0.06, 0.06, -0.16, 0.16, 0.08, -0.14, 0.46, -0.39, 0.21, -0.0, -0.08, -0.02, -0.31, -0.07, 0.16, -0.0, 0.27, -0.41, -0.19, 0.06, 0.1, -0.17, -0.51, -0.46, 0.11, -0.3, 0.21, -0.0, -0.04, -0.27, -0.0, -0.14, 0.04, 0.35, 0.21, -0.22, 0.2, -0.12, 0.1, 0.04, -0.46, 0.08, 0.01 }, {-0.06, 0.08, 0.09, 0.24, -0.13, 0.02, 0.31, -0.29, 0.09, 0.05, 0.13, 0.12, 0.08, 0.03, 0.24, 0.27, -0.14, 0.33, -0.21, 0.3, 0.23, 0.01, 0.02, 0.01, -0.61, -0.09, 0.15, -0.26, -0.18, -0.08, 0.14, 0.09, 0.27, 0.34, -0.24, 0.05, 0.02, -0.25, -0.16, -0.39, -0.26, 0.17, -0.0, 0.32, 0.06, 0.13, -0.08, 0.05, -0.33, 0.08, 0.35, 0.11, -0.13, 0.35, -0.04, 0.03, -0.1, -0.43, 0.32, -0.12 }, {-0.04, 0.48, 0.02, -0.06, -0.1, -0.21, 0.26, -0.16, 0.1, 0.07, 0.17, 0.1, 0.0, 0.22, 0.17, 0.32, -0.01, 0.24, 0.2, 0.11, 0.07, 0.11, -0.02, 0.12, -0.0, 0.01, 0.02, -0.42, 0.29, 0.17, 0.55, 0.13, 0.15, 0.11, -0.11, -0.03, 0.04, 0.09, -0.09, -0.34, 0.11, 0.18, 0.57, 0.51, -0.02, 0.04, -0.13, 0.03, 0.07, 0.01, 0.06, -0.24, -0.25, 0.1, -0.15, 0.37, 0.27, -0.05, 0.37, 0.36 }, {0.1, 0.16, 0.13, -0.04, 0.12, -0.15, -0.2, -0.19, 0.2, 0.07, 0.16, 0.02, 0.01, 0.25, -0.18, 0.01, -0.05, -0.19, 0.31, -0.26, 0.07, 0.26, 0.11, 0.03, -0.26, 0.3, -0.11, -0.04, 0.08, 0.29, 0.6, -0.1, 0.02, 0.05, -0.27, -0.13, -0.04, 0.38, 0.37, 0.34, 0.17, 0.46, 0.69, 0.59, -0.18, 0.27, -0.28, 0.22, 0.41, 0.11, -0.02, 0.07, 0.04, -0.07, 0.1, 0.12, -0.03, 0.06, 0.3, 0.17 }, {-0.07,

-0.05, -0.01, -0.05, 0.25, -0.49, -0.24, 0.04, 0.16, 0.09, 0.06, 0.09, 0.09, -0.02, -0.11, 0.22, -0.13, 0.07, 0.0, 0.33, -0.29, -0.13, 0.23, -0.17, 0.33, 0.25, 0.25, 0.28, 0.17, 0.39, 0.49, 0.08, 0.05, 0.16, 0.01, 0.4, 0.07, 0.26, 0.31, -0.01, 0.22, 0.12, 0.4, 0.34, -0.21, 0.17, 0.07, 0.22, 0.24, 0.03, -0.28, -0.11, -0.08, -0.65, -0.08, 0.39, -0.02, 0.44, 0.49, 0.01 }, {-0.07, -0.01, -0.15, -0.05, 0.02, -0.31, -0.17, -0.23, 0.1, 0.09, 0.21, 0.1, 0.09, -0.15, -0.02, 0.17, -0.28, 0.22, -0.09, 0.12, -0.54, -0.13, 0.11, -0.24, 0.19, 0.09, -0.01, 0.2, 0.11, 0.27, 0.69, 0.19, 0.05, 0.22, 0.12, 0.3, -0.04, 0.15, 0.27, 0.03, 0.37, 0.18, 0.52, 0.46, -0.06, 0.21, 0.24, 0.05, -0.05, 0.04, -0.09, -0.28, 0.08, -0.25, -0.19, 0.14, 0.19, 0.35, 0.72, 0.11 }, {0.31, -0.17, -0.01, 0.11, -0.2, -0.2, 0.02, -0.3, 0.05, 0.05, 0.35, 0.02, 0.05, 0.24, 0.11, 0.38, -0.23, 0.2, 0.09, -0.34, -0.35, -0.18, 0.21, -0.27, 0.18, 0.22, 0.04, 0.06, -0.19, 0.09, 0.21, 0.21, -0.1, 0.44, 0.14, -0.03, -0.04, 0.2, 0.22, 0.09, 0.0, 0.21, 0.64, 0.13, -0.21, 0.19, 0.11, 0.12, 0.14, 0.0, -0.23, 0.05, -0.1, -0.48, -0.24, 0.36, 0.24, 0.12, 0.46, 0.07 }, {0.45, 0.11, -0.17, 0.11, -0.41, -0.01, 0.21, -0.06, -0.13, 0.05, 0.54, 0.05, 0.03, -0.05, 0.15, -0.12, -0.17, 0.26, -0.02, 0.08, -0.37, 0.2, 0.36, 0.21, 0.01, -0.04, -0.22, 0.09, -0.01, 0.22, 0.34, 0.48, -0.25, 0.27, -0.1, 0.22, 0.03, -0.38, -0.01, -0.17, -0.03, 0.19, 0.39, 0.34, 0.05, 0.18, -0.05, -0.03, 0.19, 0.03, -0.13, 0.23, 0.16, 0.09, -0.07, 0.04, -0.11, 0.03, 0.1, 0.21 }, {0.31, -0.31, 0.16, 0.38, -0.41, 0.57, 0.42, -0.26, 0.01, 0.09, -0.28, 0.07, 0.2, -0.29, -0.24, -0.06, -0.19, -0.36, -0.07, -0.34, 0.32, 0.03, 0.02, 0.09, -0.33, -0.03, -0.02, -0.02, -0.28, -0.18, 0.12, -0.18, -0.19, 0.46, -0.17, 0.01, 0.06, 0.06, -0.33, 0.21, -0.42, 0.13, 0.12, 0.19, -0.29, 0.07, 0.13, -0.15, -0.05, 0.21, -0.15, 0.38, 0.09, 0.15, 0.08, -0.07, 0.13, -0.59, -0.07, 0.19 }, {0.46, -0.22, 0.36, 0.22, -0.42, 0.56, 0.3, -0.27, 0.01, 0.02, -0.27, 0.03, 0.17, -0.08, -0.09, 0.02, -0.03, -0.21, -0.29, -0.19, 0.33, 0.42, 0.23, 0.45, 0.0, -0.14, 0.05, 0.05, -0.08, -0.22, -0.22, -0.09, -0.12, 0.42, -0.26, 0.03, 0.22, -0.29, -0.56, 0.19, -0.45, 0.25, 0.2, -0.06, -0.27, 0.13, 0.11, -0.06, -0.07, 0.22, -0.05, 0.63, 0.01, 0.63, 0.24, -0.34, -0.2, -0.44, -0.25, 0.21 }, {0.8, -0.2, 0.14, 0.2, -0.55, 0.77, 0.63, -0.06, -0.07, -0.08,

-0.53, 0.11, 0.26, 0.05, -0.05, 0.38, -0.35, -0.43, 0.5, -0.5, -0.06, 0.33, 0.03, 0.49, -0.43, -0.19, 0.17, 0.34, -0.07, -0.45, -0.6, 0.05, 0.17, 0.62, -0.25, -0.33, 0.09, -0.46, -0.83, -0.21, -0.57, -0.12, -0.15, -0.42, -0.34, -0.06, 0.0, -0.0, 0.49, 0.26, -0.08, 0.82, 0.03, 0.29, 0.04, -0.14, 0.23, -0.37, -0.26, 0.58 }, {0.57, -0.68, 0.31, 0.24, -0.54, 0.93, 0.54, -0.48, -0.06, -0.01, -0.75, 0.04, 0.35, 0.04, -0.07, 0.37, -0.57, -0.51, 0.02, -0.69, -0.33, 0.5, 1.01, 0.53, -0.67, 0.02, -0.23, 0.37, -0.32, -0.75, -0.16, -0.08, 0.3, 0.85, -0.26, -0.19, 0.18, -0.33, -0.59, -0.07, -0.09, -0.19, 0.11, -0.11, -0.36, -0.18, 0.23, -0.17, -0.02, 0.23, -0.13, 0.97, 0.19, -0.28, 0.01, -0.1, 0.4, -0.25, 0.17, 0.44 }, {-0.12, -0.21, -0.03, -0.28, 0.23, 0.23, -0.11, -0.19, -0.18, 0.01, -0.06, 0.05, -0.07, -0.27, 0.49, -0.31, -0.6, 0.05, 0.16, 0.37, -0.0, 0.04, 0.51, 0.27, -0.11, -0.33, 0.06, 0.19, 0.35, 0.01, 0.22, 0.32, 0.03, -0.26, -0.48, 0.06, 0.11, -0.06, 0.06, -0.09, 0.18, -0.08, -0.01, -0.08, 0.61, -0.28, -0.25, -0.29, 0.16, -0.07, 0.38, -0.1, 0.22, 0.6, 0.26, 0.09, 0.14, 0.41, -0.12, 0.29 }, {-0.19, -0.1, -0.06, -0.27, 0.14, 0.23, -0.08, -0.19, -0.21, 0.09, -0.03, 0.0, -0.06, -0.33, 0.5, -0.33, -0.51, -0.05, 0.12, 0.41, 0.02, 0.02, 0.51, 0.22, -0.16, -0.3, 0.06, 0.27, 0.32, 0.04, 0.22, 0.3, 0.06, -0.33, -0.53, -0.02, 0.07, 0.0, -0.02, -0.09, 0.29, -0.06, -0.01, -0.13, 0.59, -0.32, -0.18, -0.19, 0.16, 0.01, 0.39, -0.14, 0.24, 0.66, 0.3, 0.01, 0.12, 0.38, -0.21, 0.21 }, {-0.1, -0.39, -0.03, -0.14, -0.01, -0.05, -0.31, -0.51, -0.11, 0.01, 0.38, 0.06, -0.06, -0.06, 0.35, -0.1, -0.38, 0.26, 0.16, -0.16, 0.4, 0.04, 0.06, 0.05, -0.09, -0.51, 0.18, -0.01, 0.11, -0.22, 0.17, 0.45, -0.21, 0.12, -0.37, 0.01, 0.11, -0.1, -0.21, 0.09, -0.16, -0.04, 0.12, -0.02, 0.29, -0.02, -0.3, -0.12, -0.23, 0.03, 0.21, 0.24, -0.12, 0.94, 0.31, -0.43, 0.08, -0.2, -0.21, 0.18 }, {-0.03, 0.06, 0.1, -0.05, 0.02, 0.08, -0.12, -0.49, -0.1, 0.13, 0.49, 0.02, -0.0, 0.31, 0.48, 0.48, -0.38, 0.43, 0.27, -0.72, 0.45, 0.0, -0.02, 0.03, -0.34, -0.08, 0.14, 0.02, -0.04, -0.61, 0.03, 0.41, 0.13, 0.36, -0.57, 0.02, 0.03, 0.2, -0.13, -0.13, -0.19, 0.09, 0.11, -0.26, 0.2, 0.19, -0.52, -0.39, -0.18, 0.14, 0.4, 0.32, -0.08, 0.63, 0.36, -0.01, 0.31, -0.24, -0.22, 0.45 }, {-0.31, -0.36, 0.18, -0.06, -0.11, 0.27, 0.23, -0.37, 0.13, 0.04, -0.07, 0.01, 0.0, 0.15, 0.41, 0.46, -0.69, 0.01, -0.03,

0.17, 0.03, 0.08, -0.1, 0.07, -0.28, 0.1, 0.12, -0.23, 0.29, -0.29, -0.06, 0.08, 0.05, 0.22, -0.19, 0.16, -0.0, 0.23, 0.12, -0.06, 0.15, -0.0, -0.12, -0.06, 0.0, 0.28, 0.07, 0.2, -0.63, 0.13, -0.0, -0.35, -0.18, 0.2, -0.06, 0.11, 0.29, -0.24, 0.32, -0.14 }, {-0.23, -0.15, 0.11, -0.2, -0.1, -0.16, 0.1, -0.24, 0.03, 0.03, -0.04, -0.0, 0.07, 0.03, 0.44, 0.07, -0.53, 0.05, -0.02, 0.47, 0.12, 0.16, 0.21, 0.09, -0.69, 0.0, -0.08, -0.14, 0.2, -0.33, -0.33, -0.01, -0.1, -0.06, -0.19, 0.04, -0.02, 0.11, 0.12, -0.02, 0.27, 0.03, -0.09, -0.1, 0.11, 0.14, 0.1, 0.16, -0.46, 0.05, 0.44, -0.54, 0.07, 0.29, 0.02, 0.16, 0.03, -0.07, 0.12, -0.09 }, {-0.25, -0.0, 0.12, 0.3, -0.08, -0.19, 0.28, -0.16, 0.08, 0.02, -0.04, 0.03, -0.01, 0.34, 0.07, -0.11, -0.2, -0.18, 0.15, 0.05, -0.17, 0.3, 0.23, 0.17, -0.49, -0.03, -0.21, -0.13, 0.06, -0.44, 0.42, -0.07, -0.06, 0.19, -0.02, 0.1, -0.05, 0.26, 0.03, -0.13, -0.05, 0.14, 0.43, 0.23, 0.05, 0.16, -0.09, -0.04, 0.31, -0.0, -0.13, 0.03, 0.19, 0.27, -0.19, 0.1, -0.48, -0.04, 0.39, -0.12 }, {0.36, -0.46, 0.28, 0.28, -0.1, 0.28, 0.14, 0.08, 0.16, 0.0, -0.09, 0.11, 0.06, 0.38, 0.0, -0.16, 0.02, -0.38, -0.01, -0.13, -0.18, 0.42, 0.37, 0.23, -0.87, -0.05, -0.33, 0.15, -0.35, -0.68, 0.43, -0.35, 0.04, 0.23, -0.12, -0.17, -0.01, 0.45, 0.08, 0.13, 0.05, 0.26, 0.72, 0.31, -0.11, 0.27, -0.17, 0.06, 0.03, 0.1, -0.09, -0.04, 0.47, 0.21, 0.11, -0.02, -0.73, -0.18, 0.58, -0.31 }, {-0.16, -0.37, -0.09, 0.0, -0.12, 0.02, -0.05, -0.37, 0.12, 0.05, -0.1, 0.0, 0.02, -0.3, 0.18, 0.01, -0.45, -0.07, -0.06, -0.35, -0.53, 0.14, 0.11, 0.18, -0.01, 0.02, -0.23, 0.62, -0.0, -0.26, 0.29, 0.24, -0.08, 0.13, 0.05, 0.4, -0.04, 0.08, -0.06, -0.44, 0.02, -0.08, 0.21, 0.31, -0.0, -0.03, -0.0, 0.29, -0.09, 0.0, 0.2, 0.33, 0.44, 0.2, -0.17, -0.14, -0.14, 0.22, 0.32, -0.1 }, {-0.03, -0.19, -0.04, 0.06, -0.14, -0.12, -0.01, -0.52, 0.04, 0.05, -0.07, 0.11, 0.07, -0.15, 0.24, 0.04, -0.48, 0.18, -0.0, -0.48, -0.46, 0.22, 0.11, 0.03, -0.04, 0.11, -0.27, 0.49, -0.09, -0.45, 0.48, 0.24, -0.11, 0.22, 0.02, 0.38, -0.04, 0.2, 0.12, -0.45, 0.06, 0.01, 0.26, 0.45, -0.05, -0.03, 0.09, 0.18, -0.22, 0.07, 0.24, 0.26, 0.38, 0.07, -0.21, 0.1, 0.04, 0.06, 0.47, 0.03 }, {-0.31, -0.18, -0.24, -0.05, 0.14, -0.39, -0.16, -0.5, -0.08, 0.1, 0.16, 0.1, -0.03, -0.16, 0.45, 0.14, -0.4, 0.12, 0.22, -0.65, 0.13, 0.08, 0.29, -0.16, 0.25, 0.13, -0.02, 0.31, 0.15,

-0.17, 0.31, 0.22, -0.11, 0.12, -0.04, 0.4, -0.08, 0.27, 0.26, -0.16, -0.25, 0.14, 0.11, 0.25, -0.06, -0.11, -0.11, 0.2, -0.08, 0.04, 0.31, 0.4, -0.06, -0.04, -0.2, 0.03, 0.25, 0.4, 0.05, 0.24 }, {0.1, 0.53, -0.21, -0.18, 0.01, -0.8, -0.01, 0.02, -0.15, 0.07, 0.63, 0.1, -0.05, -0.04, 0.21, -0.1, -0.24, 0.44, 0.38, -0.22, 0.05, 0.13, 0.3, -0.17, 0.12, 0.08, -0.1, 0.15, 0.3, 0.55, 0.48, 0.42, -0.3, -0.22, -0.09, 0.18, -0.06, -0.3, 0.54, -0.24, -0.25, 0.19, -0.07, 0.34, 0.09, -0.01, -0.39, 0.09, 0.27, -0.06, 0.35, 0.1, -0.11, 0.06, -0.17, 0.17, -0.16, 0.46, -0.07, 0.53 }, {0.5, 0.13, 0.09, 0.05, -0.34, -0.22, 0.53, 0.32, -0.05, 0.05, 0.28, 0.05, 0.04, -0.18, 0.01, 0.24, -0.04, 0.22, 0.26, -0.21, -0.16, -0.23, -0.03, -0.17, -0.01, 0.1, -0.27, -0.06, 0.07, 0.46, 0.1, 0.12, -0.03, 0.43, 0.4, -0.56, 0.0, -0.25, -0.04, 0.36, 0.35, 0.16, 0.27, 0.04, 0.03, 0.16, -0.08, 0.12, 0.6, 0.02, -0.36, -0.01, -0.06, -0.25, -0.06, -0.09, -0.28, 0.44, 0.17, 0.1 }, {0.09, 0.16, 0.01, 0.55, 0.08, -0.15, 0.37, 0.22, -0.12, 0.03, 0.28, 0.07, 0.02, 0.2, -0.02, 0.26, 0.17, 0.45, 0.11, 0.11, 0.16, -0.13, 0.29, 0.03, -0.12, -0.0, 0.07, -0.05, 0.09, 0.23, 0.07, 0.1, -0.24, 0.45, 0.26, -0.08, 0.13, -0.26, -0.01, -0.17, 0.19, 0.09, -0.0, 0.02, 0.2, 0.1, 0.04, -0.2, 0.2, 0.03, -0.37, -0.07, -0.1, -0.22, -0.13, 0.03, -0.47, 0.62, 0.19, -0.07 }, {-0.19, 0.01, -0.15, 0.55, 0.01, -0.08, 0.23, -0.11, -0.19, 0.02, 0.29, 0.03, -0.03, -0.21, 0.03, -0.31, -0.39, 0.14, -0.2, -0.17, -0.25, 0.1, 0.25, 0.05, -0.4, -0.26, -0.3, 0.15, 0.17, -0.3, 0.55, 0.66, -0.26, -0.02, 0.03, 0.07, -0.08, -0.6, -0.2, -0.23, 0.02, -0.04, 0.09, 0.42, 0.38, -0.19, 0.2, -0.06, 0.09, -0.06, 0.17, 0.11, 0.5, -0.02, -0.26, -0.34, 0.05, 0.04, -0.09, 0.23 }, {0.34, -0.31, 0.1, 0.21, -0.03, 0.07, -0.09, -0.18, -0.21, -0.08, 0.09, 0.08, -0.14, -0.39, -0.32, -0.67, -0.24, 0.06, 0.01, -0.82, -0.26, 0.55, 1.25, 0.23, -0.85, -0.01, -0.75, 0.37, -0.3, -0.94, 0.11, 0.42, -0.18, -0.04, -0.44, -0.03, -0.08, -0.53, 0.21, -0.23, -0.07, -0.14, 0.18, 0.14, 0.4, -0.25, 0.1, -0.12, 0.35, -0.14, 0.87, 0.59, 0.95, 0.07, -0.17, -0.27, -0.34, -0.11, -0.05, 0.67 }, {-0.19, 0.09, -0.0, -0.28, -0.07, 0.15, 0.28, -0.53, -0.23, 0.1, 0.03, 0.07, 0.08, -0.43, 0.38, -0.08, -0.31, -0.3, 0.04, -0.2, -0.5, -0.19, 0.13, -0.01, -0.07, -0.64, -0.44, 0.25, 0.47, -0.29, 0.67, 0.07, 0.17, 0.19, -0.06, 0.31, 0.06, 0.05, 0.23, -

0.17, 0.31, -0.28, 0.11, 0.25, 0.61, -0.3, -0.14, -0.5, 0.22, 0.08, 0.07, 0.13, 0.09, 0.43, -0.09, -0.51, 0.34, 0.18, 0.36, -0.21 }, {-0.2, 0.05, 0.0, -0.2, 0.02, 0.04, 0.28, -0.52, -0.29, 0.11, 0.1, 0.07, 0.08, -0.45, 0.47, -0.1, -0.28, -0.22, 0.03, -0.09, -0.5, -0.25, 0.15, 0.06, -0.02, -0.55, -0.48, 0.28, 0.56, -0.26, 0.69, 0.09, 0.18, 0.17, -0.04, 0.21, 0.07, -0.05, 0.27, -0.07, 0.33, -0.34, 0.1, 0.33, 0.64, -0.29, -0.25, -0.56, 0.22, 0.14, 0.01, 0.15, 0.15, 0.41, -0.17, -0.42, 0.26, 0.14, 0.28, -0.11 }, {-0.07, 0.11, 0.03, -0.29, 0.23, -0.48, 0.4, -0.52, -0.15, 0.0, 0.18, 0.04, 0.05, -0.34, 0.57, 0.43, -0.26, 0.25, 0.24, -0.49, -0.12, -0.2, -0.02, -0.11, -0.15, -0.07, -0.31, 0.18, 0.14, -0.01, 0.21, 0.44, -0.06, 0.08, -0.06, 0.1, 0.06, 0.53, 0.58, -0.45, 0.11, -0.06, -0.37, -0.0, 0.36, -0.16, -0.19, -0.56, 0.01, 0.09, -0.06, 0.34, -0.09, -0.42, -0.09, 0.28, 0.4, 0.08, -0.18, 0.23 }, {0.05, 0.18, 0.08, -0.3, 0.03, -0.3, 0.35, -0.39, -0.2, 0.07, 0.56, 0.07, 0.07, -0.09, 0.67, 0.57, -0.26, 0.36, 0.33, -0.97, -0.48, -0.38, -0.24, -0.17, -0.15, 0.16, -0.27, 0.02, 0.25, -0.3, 0.12, 0.31, -0.02, 0.45, -0.2, -0.05, 0.06, 0.41, 0.35, -0.39, -0.25, -0.1, -0.07, -0.13, 0.22, -0.21, -0.61, -0.39, 0.25, 0.15, -0.05, 0.75, -0.12, -0.15, -0.06, 0.56, 0.57, 0.2, -0.05, 0.06 }, {0.31, -0.05, 0.25, -0.65, 0.03, -0.03, 0.52, -0.12, 0.06, 0.09, -0.05, 0.09, 0.17, -0.18, 0.39, 0.05, -0.31, -0.26, 0.19, -0.44, -0.58, 0.38, -0.07, 0.03, -0.15, 0.17, -0.71, 0.21, 0.12, -0.48, -0.01, -0.43, -0.04, 0.15, 0.07, -0.01, 0.08, 0.26, 0.31, 0.12, 0.31, -0.25, -0.13, -0.16, 0.01, 0.05, -0.07, 0.16, 0.41, 0.12, -0.11, 0.29, 0.19, -0.4, -0.0, -0.17, -0.06, 0.14, 0.13, 0.03 }, {0.12, 0.0, 0.3, -0.52, -0.04, -0.21, 0.42, -0.1, 0.01, 0.09, -0.09, 0.1, 0.15, -0.32, 0.34, 0.15, -0.31, -0.25, -0.02, 0.04, -0.39, -0.03, 0.03, -0.03, -0.31, 0.12, -0.51, 0.26, -0.0, -0.36, 0.01, -0.25, -0.13, 0.3, 0.16, 0.11, -0.06, 0.21, 0.39, -0.02, 0.3, -0.26, -0.04, -0.12, 0.06, 0.03, -0.03, 0.05, 0.05, 0.08, -0.06, -0.05, 0.38, -0.52, -0.06, -0.02, -0.14, 0.03, 0.27, -0.31 }, {0.58, 0.4, 0.12, -0.18, 0.09, -0.06, 0.45, 0.61, 0.05, 0.07, -0.16, -0.0, 0.1, 0.18, 0.15, 0.09, 0.13, -0.39, 0.78, 0.69, 0.03, 0.09, -0.2, 0.06, -0.42, -0.33, -0.41, 0.1, 0.15, 0.36, -0.83, -0.48, -0.06, -0.26, 0.11, -0.35, -0.01, -0.27, -0.53, 0.24, 0.62, -0.16, -0.03, -0.78, -0.01, 0.18, -0.28,

0.11, 0.48, 0.06, -0.03, -0.34, 0.25, -0.18, 0.16, -0.1, -0.65, -0.06, -0.38, 0.34 }, {0.08, -0.1, 0.18, -0.03, 0.06, 0.03, 0.17, -0.23, 0.01, 0.03, -0.05, 0.07, 0.09, -0.06, 0.04, 0.13, -0.06, -0.2, 0.17, -0.1, 0.09, 0.38, 0.55, 0.24, -0.13, -0.13, -0.42, 0.56, -0.23, -0.46, 0.2, -0.32, -0.21, 0.12, -0.16, 0.13, -0.04, 0.27, -0.04, 0.18, 0.09, 0.15, 0.4, -0.05, -0.12, 0.33, -0.38, -0.03, -0.29, 0.04, -0.06, -0.05, 0.69, -0.24, 0.13, -0.32, -0.67, -0.13, 0.21, -0.13 }, {-0.01, -0.35, 0.03, 0.15, -0.03, 0.29, -0.35, 0.12, 0.12, 0.07, -0.44, 0.07, 0.02, 0.09, -0.2, 0.04, 0.01, -0.27, 0.03, -0.0, -0.09, 0.03, -0.05, 0.04, 0.28, -0.39, -0.23, 0.37, -0.13, 0.43, 0.1, -0.06, -0.22, -0.13, -0.1, 0.37, 0.03, 0.08, -0.34, 0.38, 0.2, 0.14, 0.48, -0.11, -0.29, 0.44, 0.27, 0.26, -0.53, 0.01, -0.15, 0.02, -0.13, 0.3, 0.37, -0.49, -0.36, 0.11, 0.2, -0.28 }, {0.01, -0.23, 0.12, 0.02, -0.02, 0.04, -0.22, -0.1, 0.17, 0.05, -0.28, 0.12, 0.08, 0.06, 0.08, 0.28, -0.08, 0.05, 0.24, -0.09, -0.06, 0.07, 0.18, 0.12, 0.33, 0.05, -0.16, 0.2, -0.27, 0.18, 0.4, 0.03, -0.21, 0.06, -0.07, 0.14, -0.02, 0.49, 0.04, 0.15, 0.15, 0.23, 0.51, 0.09, -0.37, 0.43, 0.17, 0.02, -0.6, 0.02, -0.13, 0.17, -0.07, -0.1, 0.33, 0.11, 0.22, 0.17, 0.35, -0.07 }, {0.31, -0.25, -0.09, -0.12, -0.32, 0.37, -0.38, -0.16, 0.11, 0.05, -0.31, 0.07, 0.15, -0.37, 0.03, 0.22, 0.19, 0.18, -0.01, -0.56, 0.29, 0.21, 0.08, -0.17, 0.05, -0.06, -0.16, -0.06, -0.32, 0.38, 0.21, -0.09, 0.1, -0.11, -0.14, -0.23, -0.02, 0.55, -0.53, 0.55, -0.0, 0.3, 0.63, -0.23, -0.4, 0.34, 0.33, 0.11, -0.44, 0.11, 0.13, 0.14, -0.25, 0.1, 0.48, -0.32, 0.17, 0.03, 0.02, 0.17 }, {-0.09, -0.07, -0.17, 0.4, -0.08, -0.15, -0.05, -0.11, 0.01, 0.06, 0.16, 0.0, 0.03, -0.47, 0.12, 0.28, -0.05, 0.18, -0.29, -0.27, 0.13, -0.09, -0.08, -0.29, 0.15, -0.07, -0.17, -0.05, 0.28, 0.3, 0.83, 0.3, 0.04, 0.04, -0.01, 0.01, -0.03, 0.21, -0.24, 0.37, -0.48, 0.27, 0.5, 0.54, -0.12, 0.24, 0.22, 0.06, -0.1, -0.01, 0.17, 0.11, -0.07, -0.19, 0.13, -0.18, -0.09, 0.38, 0.29, 0.1 }, {0.07, 0.19, -0.2, 0.32, -0.11, -0.27, 0.53, -0.44, -0.0, 0.11, 0.31, 0.02, -0.0, -0.15, 0.35, -0.0, -0.05, 0.39, -0.08, -0.47, -0.31, 0.07, -0.02, 0.22, 0.18, 0.14, -0.17, 0.04, 0.06, 0.0, 0.49, 0.15, 0.08, 0.11, -0.08, 0.17, -0.01, -0.08, -0.22, -0.01, -0.34, 0.22, 0.11, 0.53, 0.24, 0.14, -0.17, -0.12, 0.15, 0.08, 0.1, 0.03, 0.18, -0.01, -0.12, -0.25,

-0.09, 0.53, 0.51, 0.11 }, {-0.11, 0.15, -0.21, 0.41, 0.16, -0.05, 0.54, -0.35, -0.11, 0.05, 0.27, 0.0, 0.04, -0.03, 0.4, 0.18, -0.07, 0.41, -0.08, -0.26, -0.16, -0.23, -0.02, 0.13, 0.06, 0.15, 0.14, -0.0, 0.06, -0.18, 0.2, 0.28, -0.0, 0.31, 0.02, -0.06, -0.01, -0.01, -0.18, -0.42, -0.19, 0.27, -0.1, 0.33, 0.27, 0.1, -0.13, -0.18, -0.05, 0.07, 0.06, 0.11, 0.06, -0.14, -0.18, 0.32, -0.15, 0.29, 0.29, -0.04 }, {-0.54, -0.11, -0.16, 0.28, 0.14, -0.46, 0.08, -0.5, -0.26, 0.01, 0.41, 0.12, -0.13, 0.0, 0.14, -0.31, -0.14, 0.36, -0.38, -0.13, 0.07, 0.32, 0.35, -0.1, -0.07, -0.37, 0.15, 0.1, 0.09, -0.34, 0.98, 0.49, 0.04, -0.09, -0.12, 0.37, -0.15, -0.3, -0.14, -0.22, -0.18, 0.24, -0.03, 0.46, 0.35, -0.19, 0.01, -0.28, -0.53, -0.19, 0.54, 0.2, 0.31, 0.14, -0.06, -0.05, 0.19, 0.15, 0.11, 0.12 }, {-0.32, -0.39, -0.25, 0.11, 0.12, -0.4, -0.38, -0.17, -0.33, 0.03, 0.35, -0.01, -0.18, -0.15, -0.64, -1.04, 0.15, 0.04, -0.42, -0.22, -0.2, 0.66, 0.56, 0.03, -0.25, -0.46, -0.49, 0.21, 0.14, -0.72, 0.83, 0.05, -0.42, -0.45, -0.46, 0.28, -0.21, -0.66, -0.06, 0.41, 0.24, 0.17, 0.22, 0.46, 0.36, -0.26, 0.16, -0.24, -0.0, -0.23, 0.91, -0.03, 0.64, 0.28, 0.16, -0.82, -0.84, 0.16, 0.24, 0.2 }, {-0.07, -0.29, 0.12, 0.06, -0.01, -0.38, 0.16, -0.32, 0.16, 0.01, -0.2, 0.01, 0.1, -0.14, 0.12, -0.09, 0.11, -0.06, -0.24, -0.25, 0.08, 0.26, 0.39, 0.07, -0.08, -0.1, 0.03, 0.3, -0.0, -0.32, -0.03, 0.2, 0.12, 0.04, -0.05, 0.41, 0.06, -0.15, -0.19, -0.34, -0.37, -0.03, 0.06, -0.04, 0.23, -0.05, 0.51, -0.15, 0.1, 0.1, 0.06, 0.0, 0.33, 0.26, -0.18, -0.25, -0.27, -0.23, 0.08, -0.05 }, {-0.06, -0.24, 0.18, 0.05, 0.07, -0.4, 0.1, -0.36, 0.14, 0.04, -0.25, 0.03, 0.02, -0.13, 0.16, -0.06, 0.15, -0.08, -0.2, -0.25, 0.09, 0.23, 0.45, 0.08, 0.02, -0.16, 0.05, 0.3, 0.0, -0.31, 0.04, 0.21, 0.18, -0.04, -0.01, 0.43, 0.1, -0.27, -0.26, -0.32, -0.34, -0.01, -0.06, -0.02, 0.13, -0.17, 0.47, -0.16, 0.04, 0.08, -0.03, 0.04, 0.38, 0.23, -0.25, -0.25, -0.19, -0.28, 0.01, -0.14 }, {0.02, -0.09, -0.0, -0.22, 0.09, -0.3, 0.25, -0.45, 0.11, 0.05, 0.21, 0.07, 0.06, -0.2, 0.36, 0.6, -0.0, 0.24, 0.1, -0.45, 0.28, 0.01, 0.03, -0.05, 0.26, 0.48, 0.31, 0.09, 0.02, -0.05, 0.08, 0.54, 0.27, 0.1, 0.1, 0.06, 0.05, 0.65, 0.49, -0.54, -0.26, 0.04, -0.33, -0.03, -0.09, -0.06, 0.2, -0.01, 0.04, 0.07, -0.47, 0.13, -0.14, -0.51, -0.32, 0.55, 0.55, -0.01, -0.18, 0.12 }, {-0.06, -0.03, 0.16, -0.45, -0.09, -0.3, 0.16, -

0.19, 0.08, 0.04, 0.27, 0.02, 0.03, -0.2, 0.14, 0.26, 0.02, 0.18, -0.03, -0.66, -0.01, -0.2, 0.07, -0.18, 0.22, 0.45, 0.31, 0.12, 0.12, -0.24, -0.03, 0.28, 0.08, 0.16, 0.04, -0.11, 0.09, 0.44, 0.39, 0.07, -0.22, -0.09, -0.14, -0.01, -0.11, -0.21, -0.21, 0.29, 0.07, 0.15, -0.6, 0.44, -0.19, -0.09, -0.05, 0.37, 0.33, 0.13, -0.05, -0.22 }, {0.31, -0.04, 0.31, -0.61, -0.34, -0.31, 0.12, 0.08, 0.1, -0.01, 0.36, 0.03, 0.11, -0.34, 0.25, -0.04, -0.29, -0.07, 0.2, -0.05, 0.24, 0.17, 0.07, -0.07, -0.27, -0.0, -0.19, -0.08, 0.1, -0.24, 0.04, 0.15, 0.08, -0.19, -0.25, 0.04, 0.11, -0.17, -0.26, 0.42, -0.14, -0.02, -0.05, -0.25, 0.16, -0.02, -0.09, 0.27, -0.03, 0.05, 0.14, 0.23, 0.22, 0.4, 0.07, -0.18, -0.0, -0.26, -0.1, 0.25 }, {0.18, -0.13, 0.34, -0.3, -0.14, -0.23, -0.04, -0.13, 0.09, 0.09, 0.26, -0.01, 0.13, -0.36, 0.07, 0.27, -0.19, 0.13, -0.05, 0.49, 0.27, -0.23, 0.04, -0.23, -0.18, 0.04, -0.07, -0.22, -0.13, -0.2, -0.05, 0.54, 0.19, -0.02, -0.02, 0.15, 0.05, -0.14, 0.09, 0.21, 0.02, 0.02, 0.04, -0.16, -0.01, -0.1, 0.25, 0.21, -0.46, 0.15, -0.11, 0.07, 0.25, 0.22, 0.04, -0.02, 0.17, -0.54, 0.02, -0.09 }, {0.54, -0.15, 0.21, -0.11, -0.32, 0.46, 0.1, 0.31, 0.09, 0.01, 0.11, 0.07, 0.11, 0.06, 0.16, 0.35, -0.15, -0.36, 0.46, 0.22, 0.17, -0.0, -0.36, 0.45, -0.56, 0.08, -0.46, -0.02, 0.07, 0.1, -0.75, 0.17, -0.31, -0.01, -0.09, -0.68, -0.06, -0.47, -0.75, 0.32, 0.55, 0.02, -0.04, -0.72, -0.25, -0.01, 0.28, 0.57, 0.64, 0.18, -0.44, -0.17, 0.34, 0.12, 0.2, -0.0, -0.13, -0.28, -0.56, 0.35 }, {-0.03, -0.19, 0.23, -0.13, -0.12, 0.71, 0.04, -0.57, 0.18, 0.1, 0.26, 0.12, 0.02, -0.35, 0.04, 0.33, -0.43, 0.1, 0.11, -0.26, 0.28, 0.17, 0.55, 0.48, -0.53, 0.27, -0.54, 0.55, -0.0, -0.48, -0.2, -0.01, -0.09, 0.12, -0.17, -0.06, -0.1, 0.13, -0.42, -0.47, -0.11, 0.2, 0.02, 0.15, 0.05, -0.23, -0.13, 0.18, -0.01, 0.1, -0.28, 0.09, 0.87, 0.34, -0.09, 0.1, -0.34, -0.41, -0.12, 0.5 }, {-0.62, -0.34, 0.08, 0.22, 0.2, 1.03, -0.13, -0.2, 0.01, 0.1, -0.55, 0.02, 0.04, -0.34, -0.05, -0.7, -0.42, 0.11, -0.26, 0.25, 0.09, 0.03, 0.3, 0.32, 0.14, -0.55, -0.27, 0.27, 0.46, 0.37, 0.53, 0.25, -0.19, 0.14, 0.08, 0.51, -0.0, -0.42, -0.26, 0.11, 0.24, 0.2, 0.16, 0.05, 0.11, 0.08, 0.67, -0.25, -0.37, 0.01, 0.18, -0.01, 0.2, 0.83, 0.34, -0.42, -0.04, 0.12, 0.18, -0.35 }, {-0.42, -0.29, -0.04, 0.26, -0.02, 0.39, -0.0, -0.42, -0.1, -0.02, -0.16, 0.06, 0.05, -0.32, 0.11,

-0.25, -0.47, 0.26, -0.03, 0.21, -0.13, -0.2, 0.33, 0.01, 0.16, -0.21, -0.18, 0.0, 0.28, 0.66, 0.73, 0.43, -0.15, 0.2, 0.07, 0.32, -0.01, -0.25, -0.2, -0.14, 0.26, 0.26, 0.27, 0.3, 0.17, -0.01, 0.3, -0.36, -0.41, 0.07, 0.15, -0.13, 0.06, 0.25, 0.15, 0.21, 0.04, 0.24, 0.5, -0.18 }, {0.25, -0.2, -0.06, 0.12, 0.01, -0.08, -0.41, 0.56, 0.17, 0.1, -0.3, 0.07, 0.16, -0.04, -0.26, -0.27, 0.43, -0.09, 0.11, 0.23, -0.47, -0.16, 0.17, -0.24, 0.17, -0.05, -0.23, -0.16, -0.0, 0.34, 0.08, -0.23, -0.03, -0.02, 0.24, -0.2, 0.09, -0.24, -0.02, 0.39, 0.43, -0.01, 0.15, -0.3, -0.21, 0.25, 0.33, 0.11, 0.19, 0.13, -0.3, -0.28, 0.03, 0.04, 0.12, -0.03, -0.7, 0.31, 0.23, -0.15 }, {-0.1, -0.12, -0.12, 0.43, 0.15, 0.12, 0.08, 0.16, 0.19, 0.11, -0.16, 0.05, 0.06, -0.06, 0.12, -0.01, 0.08, 0.02, -0.3, 0.43, 0.05, -0.16, -0.09, -0.28, 0.2, -0.08, -0.04, -0.0, 0.52, -0.01, 0.42, -0.0, 0.08, -0.16, 0.34, 0.28, 0.03, -0.15, -0.4, -0.03, 0.05, -0.08, -0.15, 0.13, -0.17, 0.35, 0.32, 0.14, 0.08, 0.15, 0.01, -0.41, 0.04, -0.04, -0.04, -0.24, -0.36, 0.4, 0.34, 0.01 }, {-0.23, -0.04, -0.12, 0.39, 0.16, -0.16, 0.36, -0.41, -0.09, 0.09, 0.18, 0.08, 0.02, -0.01, 0.47, 0.06, -0.21, 0.19, -0.09, -0.09, -0.59, 0.01, -0.15, 0.26, 0.6, 0.16, -0.36, 0.0, 0.35, -0.18, 0.8, 0.04, 0.01, 0.14, 0.24, 0.6, -0.06, -0.08, -0.13, -0.4, 0.17, -0.09, -0.18, 0.6, 0.2, 0.08, -0.15, 0.01, 0.04, 0.03, -0.47, -0.23, 0.25, -0.3, -0.3, -0.32, 0.18, 0.5, 0.55, -0.17 }, {-0.14, 0.29, -0.14, 0.59, 0.46, -0.39, 0.13, -0.26, 0.0, 0.06, 0.18, 0.04, 0.1, 0.4, 0.28, 0.01, 0.12, 0.32, 0.07, -0.49, -0.19, -0.33, -0.11, -0.03, 0.27, 0.28, -0.03, 0.06, 0.34, -0.16, 0.39, 0.06, 0.1, 0.43, 0.06, 0.21, -0.05, 0.16, 0.07, -0.43, 0.1, 0.17, -0.2, 0.43, 0.18, 0.06, -0.42, -0.05, 0.28, 0.08, -0.16, 0.24, 0.24, -0.48, -0.22, 0.37, -0.33, 0.52, 0.46, -0.23 }, {-0.69, 0.04, -0.1, -0.01, 0.33, -0.42, -0.06, -0.22, -0.16, 0.05, 0.19, 0.04, -0.12, 0.45, 0.25, -0.19, -0.13, 0.08, -0.12, 0.07, -0.21, -0.14, 0.03, -0.11, 0.11, 0.05, 0.02, 0.07, 0.5, 0.18, 0.69, 0.42, -0.01, -0.15, 0.3, 0.28, -0.11, -0.09, 0.22, -0.09, 0.32, 0.05, -0.17, 0.35, 0.12, -0.26, -0.09, 0.09, 0.15, -0.16, 0.29, 0.15, 0.16, -0.01, -0.08, -0.06, -0.01, 0.58, 0.17, 0.1 }, {-0.81, -0.12, -0.1, 0.07, 0.55, -0.3, -0.4, 0.02, -0.26, -0.0, 0.06, 0.04, -0.14, 0.38, -0.2, -0.73, 0.13, -0.08, -0.17, 0.2, -0.1, 0.19, 0.14, -0.2, -0.03, -0.01, -0.24,

0.1, 0.51, -0.18, 0.55, -0.17, -0.35, -0.37, 0.11, 0.48, -0.24, -0.33, 0.13, 0.14, 0.69, 0.01, -0.19, 0.31, 0.1, -0.31, -0.01, 0.08, 0.18, -0.17, 0.36, -0.19, 0.12, -0.14, 0.1, -0.51, -0.69, 0.67, 0.16, -0.08 }, {-0.08, 0.01, 0.2, 0.2, 0.15, -0.13, -0.05, -0.22, 0.17, 0.11, -0.55, 0.11, 0.04, 0.05, 0.12, 0.27, 0.25, -0.14, -0.08, 0.06, 0.32, 0.15, 0.23, 0.09, 0.15, 0.02, 0.23, 0.07, -0.29, -0.15, -0.17, -0.1, 0.14, -0.09, -0.0, 0.0, 0.14, 0.03, -0.02, -0.39, -0.21, 0.04, -0.1, 0.15, -0.08, 0.13, 0.13, -0.06, 0.03, 0.08, 0.08, -0.15, -0.11, -0.0, -0.0, -0.04, 0.43, -0.22, 0.1, -0.01 }, {0.07, 0.16, 0.04, 0.08, 0.0, -0.13, -0.02, -0.34, 0.2, 0.1, -0.4, 0.02, 0.06, 0.03, 0.04, 0.24, 0.12, -0.15, -0.07, 0.04, 0.25, 0.16, 0.14, 0.06, 0.18, 0.1, 0.18, 0.1, -0.25, -0.1, -0.2, -0.15, 0.14, -0.06, -0.01, -0.08, 0.07, 0.05, -0.02, -0.22, -0.32, -0.02, -0.08, 0.19, 0.01, 0.07, 0.05, 0.04, 0.09, 0.13, -0.01, -0.12, -0.13, -0.02, -0.07, 0.04, 0.52, -0.25, 0.07, -0.06 }, {0.24, 0.2, 0.01, -0.16, 0.13, -0.2, 0.01, -0.09, 0.1, 0.11, -0.25, 0.03, 0.15, 0.17, 0.19, 0.28, 0.14, 0.01, 0.38, -0.05, 0.42, 0.11, 0.1, -0.06, 0.3, 0.44, 0.34, 0.09, -0.03, 0.05, -0.51, -0.1, 0.08, -0.2, 0.2, -0.24, 0.1, 0.49, 0.41, -0.12, -0.01, -0.12, -0.36, 0.05, -0.09, 0.05, -0.03, 0.24, 0.35, 0.1, -0.32, 0.21, -0.44, -0.4, -0.24, 0.38, 0.22, 0.06, -0.36, 0.08 }, {0.33, 0.34, 0.2, -0.44, 0.14, 0.23, 0.03, 0.5, 0.01, 0.08, 0.01, 0.05, 0.05, -0.16, 0.3, -0.37, -0.05, -0.48, 0.3, -0.06, 0.26, 0.13, 0.05, -0.03, 0.0, 0.35, 0.42, 0.23, -0.16, -0.26, -0.86, -0.17, -0.2, -0.39, 0.03, -0.39, 0.02, 0.01, 0.13, 0.48, -0.01, -0.25, -0.43, -0.19, 0.02, -0.06, 0.0, 0.25, 0.42, 0.03, -0.19, 0.15, -0.33, 0.26, 0.11, -0.03, -0.05, 0.27, -0.44, 0.07 }, {-0.09, -0.22, 0.35, -0.21, -0.15, 0.24, 0.33, -0.11, 0.06, 0.06, 0.01, 0.09, 0.07, -0.5, 0.57, 0.24, -0.3, 0.03, -0.04, -0.12, 0.36, -0.12, 0.03, -0.02, 0.09, 0.2, 0.18, 0.24, -0.14, -0.11, 0.15, 0.17, 0.4, 0.12, 0.21, 0.01, 0.09, 0.0, 0.12, 0.12, -0.02, -0.12, 0.07, -0.03, -0.07, 0.11, 0.13, 0.05, -0.15, 0.07, -0.34, -0.02, 0.0, -0.21, -0.19, 0.22, 0.35, 0.05, 0.15, -0.08 }, {-0.21, -0.49, 0.35, -0.21, -0.08, -0.07, -0.19, -0.31, 0.12, 0.12, -0.23, 0.11, 0.12, -0.43, 0.13, 0.25, 0.0, 0.18, -0.73, 0.07, 0.12, -0.13, 0.32, -0.46, -0.01, 0.39, 0.06, 0.09, -0.7, 0.04, 0.41, 0.11, 0.28, -0.19, 0.16, 0.0, 0.11, -0.26, 0.64, 0.21, 0.29,

-0.03, 0.2, 0.5, -0.2, 0.02, 0.55, -0.08, -0.92, 0.08, -0.01, -0.39, 0.07, -0.28, -0.09, -0.09, 0.31, -0.36, 0.36, -0.16 }, {0.12, -0.27, 0.11, -0.27, -0.5, 0.6, 0.06, -0.05, 0.12, 0.09, -0.58, 0.07, 0.04, -0.01, 0.28, -0.1, 0.24, -0.47, 0.12, 0.14, 0.11, 0.42, -0.25, 0.41, -0.73, 0.2, -0.32, 0.38, -0.35, -0.46, -0.83, 0.01, 0.16, 0.05, 0.08, -0.65, 0.1, -0.54, -0.67, 0.26, 0.22, -0.09, -0.12, -0.3, -0.3, -0.24, 0.16, 0.37, 0.26, 0.06, 0.14, -0.01, 0.17, -0.08, 0.35, -0.22, 0.06, -0.14, -0.31, 0.37 }, {-0.03, -0.4, 0.07, 0.09, -0.36, 1.02, 0.0, -0.22, 0.14, 0.06, -0.54, 0.06, 0.08, -0.21, -0.12, -0.18, 0.1, -0.08, 0.06, -0.05, -0.18, 0.32, 0.32, 0.5, -0.52, 0.28, -0.31, 0.63, -0.17, -0.35, -0.69, -0.56, 0.25, 0.04, 0.37, -0.12, -0.02, -0.22, -0.66, -0.28, 0.17, -0.1, -0.16, -0.0, -0.08, -0.2, -0.22, 0.11, 0.23, 0.01, -0.52, -0.14, 0.48, 0.01, -0.13, -0.26, -0.55, -0.36, 0.19, 0.09 }, {-0.49, -0.7, 0.23, 0.1, 0.12, 1.5, -0.48, -0.24, 0.14, 0.09, -1.05, 0.04, 0.01, -0.23, -0.59, -0.98, 0.05, -0.0, -0.61, 0.05, -0.35, 0.17, 0.22, 0.29, 0.34, -0.23, -0.46, -0.1, 0.07, 0.3, 0.61, -0.91, 0.15, 0.21, 0.15, 0.31, 0.18, -0.36, -0.06, 0.71, 1.02, 0.27, 0.38, 0.33, -0.05, 0.33, 0.52, -0.28, -0.07, 0.06, -0.31, -0.37, 0.18, 0.13, 0.45, -0.57, -0.61, 0.17, 0.71, -1.1 }, {-0.25, -0.29, -0.1, 0.24, 0.02, 0.83, -0.47, -0.03, 0.05, 0.06, -0.45, 0.13, 0.01, -0.02, -0.37, -0.49, 0.15, 0.25, -0.21, 0.25, -0.65, -0.11, -0.19, 0.01, 0.32, -0.29, -0.2, -0.33, 0.15, 0.9, 0.75, -0.53, 0.09, 0.18, 0.27, 0.22, 0.16, -0.46, -0.29, 0.25, 0.69, 0.19, 0.37, 0.29, 0.16, 0.2, 0.09, -0.09, -0.03, 0.08, -0.45, -0.44, -0.06, -0.13, 0.26, -0.2, -0.5, 0.29, 0.68, -0.82 }, {-0.3, 0.03, 0.01, -0.05, 0.32, -0.31, -0.52, 0.59, 0.13, 0.06, -0.54, 0.02, 0.14, -0.34, -0.13, -0.42, 0.41, -0.43, 0.29, 0.4, -0.48, -0.29, 0.16, -0.39, 0.22, -0.08, 0.02, -0.12, 0.15, 0.18, 0.16, -0.35, 0.08, -0.51, 0.31, -0.1, 0.11, -0.11, 0.39, 0.4, 0.46, -0.14, -0.19, 0.16, -0.18, -0.13, 0.09, 0.41, 0.67, 0.06, -0.34, -0.4, -0.06, -0.44, 0.01, 0.21, -0.47, 0.74, 0.24, -0.46 }, {-0.15, -0.12, 0.02, 0.71, 0.13, 0.18, 0.12, 0.61, 0.31, 0.05, -0.14, 0.04, 0.11, 0.5, 0.02, -0.06, 0.02, -0.41, 0.17, 0.57, -0.26, -0.46, -0.04, -0.03, -0.04, 0.04, -0.25, 0.04, 0.35, -0.18, 0.25, -0.24, 0.19, 0.06, 0.52, -0.0, 0.11, -0.24, -0.03, -0.35, 0.27, -0.16, -0.26, 0.13, -0.27, 0.17, 0.19, 0.35, 0.64, 0.17,

-0.43, -0.39, 0.39, -0.64, -0.41, 0.67, -0.01, 0.44, 0.24, -0.53 }, {-0.03, 0.23, -0.15, 0.66, 0.28, -0.3, 0.31, -0.12, 0.02, 0.04, 0.34, 0.02, 0.11, 0.47, 0.07, 0.15, 0.03, 0.27, 0.1, -0.23, -0.15, -0.19, -0.12, 0.1, 0.12, -0.25, -0.42, -0.07, 0.07, -0.08, 0.58, 0.1, 0.29, 0.1, 0.38, 0.31, -0.1, -0.44, -0.28, -0.71, 0.39, 0.02, -0.15, 0.37, 0.21, -0.04, -0.25, -0.09, 0.29, 0.16, -0.29, -0.4, 0.37, -0.42, -0.45, 0.22, 0.06, 0.1, 0.38, -0.02 }, {-0.28, 0.25, -0.19, 0.46, 0.33, -0.38, -0.02, -0.03, 0.05, 0.07, 0.26, 0.07, 0.0, 0.48, 0.03, -0.19, 0.26, 0.28, 0.09, -0.63, -0.16, -0.15, -0.21, -0.07, 0.09, -0.09, -0.22, 0.07, 0.24, -0.06, 0.44, -0.08, 0.22, 0.07, -0.02, 0.36, -0.03, -0.1, -0.02, -0.28, 0.14, 0.27, -0.03, 0.61, 0.16, 0.02, -0.42, -0.21, 0.21, -0.01, -0.01, 0.28, 0.25, -0.13, -0.28, 0.21, -0.5, 0.28, 0.41, -0.18 }, {-0.44, 0.29, -0.2, -0.03, 0.24, -0.65, -0.29, 0.13, -0.21, -0.01, 0.47, 0.08, -0.15, 0.63, -0.17, -0.1, 0.23, 0.36, 0.04, 0.23, 0.28, -0.1, 0.01, -0.42, 0.16, 0.14, 0.13, -0.24, 0.26, 0.36, 0.54, 0.05, -0.12, -0.46, 0.02, 0.21, -0.1, -0.05, 0.17, -0.11, 0.27, 0.14, 0.08, 0.42, 0.06, -0.19, -0.3, 0.07, -0.36, -0.13, -0.03, 0.02, -0.04, 0.29, -0.26, -0.1, 0.16, 0.13, 0.32, -0.17 }, {-0.58, 0.34, -0.07, -0.22, 0.54, -0.54, -0.29, 0.47, -0.29, -0.0, 0.34, 0.1, -0.18, 0.63, -0.67, -0.47, 0.39, 0.1, 0.01, 0.55, 0.37, -0.16, -0.24, -0.56, 0.06, 0.14, -0.21, -0.44, 0.43, 0.54, 0.58, -0.3, -0.48, -0.39, 0.11, 0.2, -0.19, -0.29, 0.04, 0.24, 0.95, 0.09, -0.01, 0.33, -0.07, -0.3, -0.18, 0.22, -0.02, -0.15, -0.16, -0.42, -0.18, -0.13, -0.16, -0.33, -0.26, 0.35, 0.2, -0.38 }, {-0.02, -0.07, 0.19, -0.02, 0.12, -0.16, -0.11, -0.09, 0.16, 0.11, -0.23, 0.09, 0.07, 0.17, -0.08, 0.26, 0.41, -0.29, 0.03, 0.17, 0.49, -0.28, 0.09, -0.33, 0.24, 0.39, 0.3, -0.24, -0.31, 0.04, -0.35, -0.28, 0.09, 0.08, 0.12, -0.02, 0.16, 0.51, 0.54, -0.06, -0.41, 0.12, 0.03, 0.01, -0.37, 0.2, 0.03, 0.07, -0.19, 0.12, -0.24, 0.07, -0.42, -0.24, -0.0, 0.4, 0.19, -0.37, 0.26, -0.17 }, {0.06, -0.09, 0.12, -0.03, -0.04, -0.23, -0.05, -0.1, 0.16, 0.07, -0.07, 0.06, 0.14, 0.18, 0.01, 0.31, 0.37, -0.25, 0.09, 0.15, 0.4, -0.34, -0.11, -0.35, 0.46, 0.3, 0.29, -0.31, -0.32, 0.11, -0.35, -0.21, 0.07, 0.1, 0.12, -0.12, 0.12, 0.42, 0.42, 0.07, -0.38, 0.18, 0.05, -0.04, -0.33, 0.23, -0.01, 0.06, -0.1, 0.14, -0.33, -0.02, -0.45, -0.18, 0.04, 0.42, 0.35,

-0.31, 0.19, -0.17 }, {-0.02, -0.25, 0.01, -0.28, -0.08, 0.04, -0.13, -0.12, 0.18, 0.05, -0.02, 0.06, 0.09, 0.33, 0.08, 0.29, 0.15, -0.03, 0.06, -0.07, 0.45, -0.23, 0.09, -0.18, 0.36, 0.55, 0.52, -0.25, -0.36, 0.06, -0.01, 0.11, 0.38, -0.01, 0.06, -0.16, 0.2, 0.84, 0.47, 0.16, -0.46, 0.17, -0.04, 0.24, -0.36, 0.29, 0.05, -0.05, -0.41, 0.15, -0.28, 0.3, -0.64, -0.33, -0.04, 0.41, 0.41, -0.51, 0.11, -0.23 }, {0.35, 0.04, 0.16, -0.61, -0.14, 0.23, -0.04, -0.07, 0.07, 0.08, 0.23, -0.01, 0.11, 0.19, 0.05, -0.03, 0.03, -0.19, 0.0, -0.06, 0.19, -0.2, 0.03, -0.11, 0.01, 0.57, 0.34, -0.34, -0.27, 0.1, -0.36, 0.02, 0.2, -0.16, -0.06, -0.3, 0.08, 0.33, 0.32, 0.62, -0.27, -0.1, 0.06, -0.05, -0.31, 0.02, 0.16, 0.08, -0.4, 0.11, -0.11, 0.44, -0.65, 0.25, 0.18, 0.18, 0.1, -0.44, -0.03, 0.01 }, {0.02, -0.08, 0.52, -0.52, 0.31, -0.12, 0.13, -0.01, 0.11, 0.03, -0.15, 0.09, 0.16, -0.26, 0.67, 0.08, -0.16, -0.32, -0.18, -0.0, 0.09, 0.22, 0.13, 0.1, 0.15, 0.46, 0.06, 0.36, -0.34, -0.24, -0.05, -0.3, 0.23, 0.04, -0.03, -0.14, 0.16, 0.46, 0.47, 0.3, -0.08, -0.04, -0.21, 0.16, -0.17, 0.11, 0.04, 0.24, 0.04, 0.17, -0.07, 0.14, 0.13, -0.18, 0.28, 0.13, 0.01, 0.24, 0.12, 0.03 }, {0.2, 0.01, 0.14, -0.58, 0.13, -0.03, -0.19, -0.02, 0.06, 0.04, -0.22, 0.04, 0.06, -0.38, 0.21, -0.05, -0.06, -0.44, -0.31, 0.06, -0.36, 0.01, -0.22, -0.22, 0.09, 0.57, -0.32, 0.32, -0.32, 0.1, -0.35, -0.1, 0.12, -0.07, -0.07, -0.26, 0.06, -0.02, 0.61, 0.33, 0.2, 0.08, -0.28, 0.28, -0.35, -0.17, 0.19, 0.4, -0.32, 0.18, 0.14, -0.17, 0.37, -0.14, 0.03, 0.2, 0.12, -0.08, 0.07, -0.0 }, {0.42, -0.21, -0.09, -0.55, -0.3, 0.7, 0.24, -0.75, 0.21, 0.03, -0.18, 0.09, 0.11, -0.57, 0.24, 0.02, -0.31, -0.2, -0.17, -0.62, -0.65, 0.63, -0.16, 0.8, -0.72, 0.43, -0.73, 0.36, -0.59, -0.07, -0.24, 0.06, -0.2, 0.02, -0.03, -0.66, 0.11, 0.4, -0.26, 0.24, 0.33, 0.32, 0.09, 0.12, -0.5, 0.12, 0.18, 0.5, -0.36, 0.05, -0.07, 0.38, 0.7, 0.09, 0.42, 0.14, 0.25, -0.11, 0.04, 0.36 }, {-0.1, -0.57, -0.22, 0.04, -0.43, 0.71, 0.2, -0.75, 0.3, 0.04, -0.47, 0.09, 0.15, -0.66, -0.21, -0.28, -0.37, -0.04, -0.4, -0.48, -0.07, 1.02, 0.14, 0.91, -0.57, -0.25, -0.5, 0.36, -0.34, -0.47, -0.25, -0.48, 0.12, -0.03, 0.05, -0.09, 0.12, 0.48, -0.8, -0.3, -0.22, 0.26, -0.04, 0.46, -0.05, 0.23, -0.04, 0.25, -0.55, 0.09, -0.3, 0.58, 0.53, 0.29, 0.18, -0.39, -0.12, -0.38, -0.04, 0.18 }, {-0.71, -0.75, 0.38, 0.49,

0.27, 1.15, -0.36, -0.0, 0.31, 0.06, -0.93, 0.05, 0.12, -0.51, -0.25, -1.32, -0.32, -0.9, -0.58, 0.54, -0.26, 0.68, 0.1, 0.92, -0.15, -0.82, -0.54, 0.18, 0.36, -0.44, 0.32, -0.79, 0.21, -0.23, 0.0, 0.55, 0.39, -0.73, -0.38, 0.39, 0.64, 0.11, 0.02, 0.16, 0.19, 0.28, 0.68, -0.02, 0.67, 0.14, 0.05, -0.26, 0.97, 0.26, 0.49, -0.73, -1.02, 0.41, 0.16, -0.56 }, {-0.59, -0.33, 0.02, 0.68, 0.08, 0.6, -0.23, 0.05, 0.13, 0.12, -0.45, 0.07, 0.14, -0.35, 0.03, -0.74, -0.34, -0.57, -0.14, 0.77, -0.49, 0.28, -0.36, 0.71, -0.09, -0.87, -0.35, 0.04, 0.66, 0.15, 0.33, -0.32, 0.22, -0.3, 0.11, 0.48, 0.3, -0.67, -0.79, -0.15, 0.25, -0.04, -0.07, 0.23, 0.41, 0.06, 0.14, 0.19, 0.76, 0.14, -0.01, -0.42, 0.45, 0.27, 0.27, -0.42, -0.6, 0.66, 0.11, -0.31 }, {-0.57, 0.1, 0.09, 0.17, 0.63, -0.89, -0.41, -0.14, 0.15, 0.11, -0.2, 0.03, 0.14, 0.15, 0.36, -0.13, 0.05, -0.15, 0.22, 0.52, -0.61, -0.03, 0.03, -0.02, 0.34, 0.31, -0.34, 0.02, 0.1, -0.13, 0.63, -0.29, -0.04, -0.24, 0.27, 0.35, 0.21, 0.11, 0.52, 0.0, 0.24, -0.03, -0.07, 0.56, 0.08, -0.0, -0.01, 0.32, 0.42, 0.15, -0.23, -0.22, 0.37, -0.41, -0.1, 0.34, -0.31, 0.97, 0.34, -0.46 }, {-0.42, 0.01, 0.02, 0.21, 0.32, -0.38, 0.06, 0.24, 0.21, 0.09, -0.27, 0.12, 0.19, -0.47, 0.12, -0.19, -0.01, -0.54, -0.1, 0.72, -0.44, -0.06, -0.23, 0.16, 0.25, 0.27, -0.33, 0.09, 0.07, -0.56, 0.89, -0.31, -0.08, 0.11, 0.15, 0.37, 0.13, -0.11, 0.17, 0.01, 0.09, 0.09, 0.2, 0.37, -0.08, 0.23, 0.33, 0.23, 0.13, 0.2, 0.1, -0.42, 0.51, -0.25, 0.13, 0.38, 0.25, 0.37, 0.49, -0.43 }, {0.09, 0.15, -0.03, -0.04, -0.02, -0.4, 0.02, 0.09, -0.01, 0.09, 0.14, 0.05, 0.08, 0.07, 0.09, -0.26, 0.11, -0.16, 0.25, 0.04, -0.25, 0.02, -0.16, 0.01, 0.04, 0.03, -0.52, 0.07, -0.03, -0.17, 0.37, -0.05, 0.18, -0.25, 0.07, 0.02, 0.0, -0.59, 0.12, -0.33, 0.48, -0.01, 0.01, 0.07, -0.0, -0.01, -0.12, 0.05, 0.11, 0.14, 0.31, -0.37, 0.28, -0.25, -0.11, 0.18, -0.01, 0.15, 0.16, 0.41 }, {-0.14, 0.15, 0.0, 0.22, -0.15, -0.3, -0.23, 0.06, -0.02, 0.05, -0.06, 0.07, 0.09, 0.17, -0.18, -0.41, 0.32, 0.29, 0.09, -0.11, -0.29, 0.14, 0.0, 0.04, 0.13, -0.23, -0.33, -0.09, 0.07, 0.14, 0.25, -0.35, 0.26, -0.29, -0.36, 0.42, 0.01, -0.51, -0.01, 0.15, 0.18, 0.36, 0.12, 0.4, 0.13, 0.02, -0.2, -0.31, 0.01, 0.09, 0.58, 0.01, 0.18, 0.19, 0.22, -0.09, -0.42, 0.02, 0.44, 0.19 }, {-0.26, 0.26, -0.11, 0.46, 0.24, -0.69, -0.14, 0.43, -0.19, -0.06, 0.16, 0.08,

-0.16, 0.29, -0.19, 0.3, 0.21, 0.36, 0.28, 0.54, -0.19, -0.26, -0.04, -0.31, 0.15, 0.2, 0.15, -0.15, 0.39, 0.58, 0.19, -0.33, -0.15, -0.53, 0.11, 0.33, -0.1, -0.26, 0.18, -0.1, 0.38, 0.16, -0.18, 0.29, 0.11, -0.24, -0.14, -0.09, 0.21, -0.11, -0.13, -0.23, -0.0, -0.48, -0.54, 0.43, -0.04, 0.48, 0.15, -0.21 }, {0.08, 0.02, 0.11, 0.46, 0.36, -0.34, -0.12, 0.69, -0.32, -0.1, -0.09, 0.08, -0.19, 0.22, -0.53, 0.1, 0.6, 0.06, 0.19, 0.22, -0.17, -0.61, 0.03, -0.46, -0.13, 0.33, -0.07, -0.23, 0.15, 0.19, 0.17, -0.67, -0.49, -0.33, 0.31, 0.15, -0.16, -0.37, 0.08, -0.09, 0.69, 0.14, -0.19, -0.03, -0.04, -0.23, 0.04, -0.11, 0.37, -0.1, -0.16, -0.57, 0.06, -1.0, -0.43, 0.57, -0.34, 0.39, 0.16, -0.45 }, {0.29, -0.12, 0.32, -0.08, -0.16, -0.08, 0.03, -0.31, 0.27, 0.1, -0.17, 0.03, 0.17, 0.15, -0.27, 0.4, 0.34, 0.2, 0.08, -0.45, 0.18, -0.15, 0.07, -0.04, 0.19, 0.44, 0.08, -0.43, -0.47, -0.3, 0.01, -0.12, 0.08, 0.27, -0.0, -0.21, 0.3, 0.38, 0.55, 0.11, -0.43, 0.27, 0.54, 0.23, -0.25, 0.34, -0.17, -0.03, -0.24, 0.12, -0.36, 0.37, -0.48, -0.37, 0.05, 0.53, 0.29, -0.24, 0.44, -0.34 }, {0.24, -0.06, 0.28, -0.08, -0.14, 0.02, 0.13, -0.2, 0.25, 0.07, -0.12, 0.04, 0.1, 0.17, -0.22, 0.38, 0.36, 0.24, 0.07, -0.49, 0.16, -0.18, 0.03, -0.09, 0.18, 0.41, 0.09, -0.39, -0.56, -0.29, 0.08, -0.19, 0.04, 0.37, 0.07, -0.25, 0.22, 0.43, 0.54, 0.16, -0.4, 0.32, 0.64, 0.32, -0.32, 0.33, -0.27, -0.04, -0.26, 0.13, -0.28, 0.26, -0.5, -0.3, 0.11, 0.53, 0.23, -0.2, 0.33, -0.24 }, {0.42, -0.3, 0.32, 0.1, -0.04, 0.11, 0.13, -0.34, 0.19, 0.0, -0.25, 0.1, 0.16, 0.59, 0.21, 0.43, 0.2, 0.31, 0.1, -0.52, 0.57, -0.1, 0.39, -0.11, -0.53, 0.6, 0.38, -0.28, -0.64, -0.29, 0.11, 0.04, 0.51, 0.18, 0.04, -0.26, 0.2, 0.55, 0.41, -0.34, -0.33, 0.09, 0.23, 0.21, -0.19, 0.25, -0.14, -0.4, -0.36, 0.13, 0.04, 0.48, -0.67, -0.65, -0.06, 0.77, 0.61, -0.52, 0.13, -0.24 }, {0.53, 0.02, 0.37, -0.2, -0.03, 0.28, -0.06, -0.45, 0.16, 0.04, 0.16, 0.1, 0.05, 0.07, 0.12, 0.25, 0.13, 0.47, 0.05, -0.86, 0.21, -0.4, 0.34, -0.22, -0.34, 0.59, 0.13, -0.08, -0.49, -0.56, 0.11, 0.23, 0.12, 0.05, -0.24, -0.32, 0.18, 0.36, 0.32, 0.1, -0.28, 0.04, 0.51, 0.36, -0.19, 0.05, -0.14, -0.46, -0.21, 0.03, 0.23, 0.5, -0.53, -0.3, 0.07, 0.53, 0.43, -0.5, 0.13, 0.12 }, {0.23, -0.14, 0.66, 0.1, 0.19, -0.1, 0.14, -0.39, 0.19, 0.06, -0.18, 0.07, 0.05, -0.03, 0.41, 0.5, 0.07, -0.14, -0.17, -0.65, 0.08, 0.36, 0.02, 0.21,

-0.23, 0.42, -0.13, 0.41, -0.56, -0.75, 0.09, -0.17, 0.16, 0.72, -0.42, -0.23, 0.18, 0.43, 0.2, -0.18, -0.7, 0.01, 0.35, 0.52, -0.18, 0.11, -0.17, -0.04, -0.06, 0.09, -0.02, 0.5, 0.25, -0.41, 0.06, 0.36, 0.27, -0.33, 0.15, -0.06 }, {0.17, 0.08, 0.16, -0.16, 0.12, -0.04, -0.3, 0.1, 0.11, 0.03, -0.25, 0.05, 0.11, -0.23, 0.01, 0.28, 0.07, -0.59, -0.26, 0.15, -0.12, 0.05, -0.35, -0.26, -0.2, 0.18, -0.23, 0.5, -0.22, 0.3, -0.57, 0.11, 0.16, 0.22, -0.39, -0.21, 0.16, -0.44, 0.32, 0.02, -0.06, 0.03, -0.23, 0.27, -0.18, -0.24, -0.01, 0.22, -0.07, 0.08, 0.11, -0.39, 0.15, -0.35, -0.01, 0.25, 0.12, -0.32, -0.06, 0.06 }, {0.44, 0.17, -0.33, -0.19, -0.09, 0.41, 0.18, -0.27, 0.19, 0.02, -0.2, 0.11, 0.11, -0.1, 0.32, -0.34, 0.12, -0.74, 0.23, 0.19, -0.29, 0.71, -0.58, 0.89, -0.62, 0.1, -0.36, 0.35, -0.36, 0.6, -0.75, 0.23, 0.06, 0.06, -0.03, -0.48, 0.13, -0.16, -0.52, 0.18, -0.2, 0.16, -0.38, -0.06, 0.02, -0.12, -0.23, 0.32, 0.37, 0.14, 0.07, 0.2, 0.15, 0.17, 0.48, 0.17, 0.28, 0.11, -0.31, 0.5 }, {0.17, -0.7, -0.05, 0.02, -0.21, 0.14, 0.11, -0.16, 0.34, 0.1, -0.34, 0.02, 0.17, -0.3, -0.37, -0.65, 0.14, -0.51, -0.19, 0.08, 0.13, 1.08, 0.08, 0.79, -0.32, -0.06, -0.51, 0.29, -0.36, -0.23, -0.37, -0.44, 0.05, -0.48, -0.11, -0.25, 0.24, 0.16, -0.71, -0.17, -0.26, 0.15, -0.25, 0.31, 0.37, 0.12, 0.02, 0.15, -0.19, 0.11, 0.15, 0.22, 0.7, 0.34, 0.44, -0.51, -0.38, 0.05, -0.31, 0.52 }, {-0.15, -0.71, 0.71, 0.06, -0.06, 0.76, -0.34, 0.27, 0.26, 0.14, -0.64, 0.04, 0.1, -0.13, -0.19, -0.94, -0.11, -1.08, -0.27, 0.85, -0.39, 0.76, 0.1, 0.57, -0.18, -0.57, -0.83, -0.04, 0.45, -0.5, -0.23, -0.72, -0.06, -0.26, -0.07, 0.08, 0.4, -0.51, -0.13, 0.53, 0.88, 0.09, 0.25, -0.19, 0.26, 0.09, 0.55, 0.1, 0.52, 0.17, 0.3, -0.54, 0.93, 0.1, 0.45, -0.63, -1.09, 0.67, 0.15, -0.21 }, {0.01, -0.41, 0.41, -0.01, -0.23, -0.0, -0.35, 0.46, 0.15, 0.06, -0.16, 0.13, 0.12, 0.09, -0.05, -0.22, -0.01, -0.76, 0.05, 1.02, -0.45, 0.27, -0.36, 0.05, -0.04, -0.57, -0.59, -0.23, 0.55, 0.53, -0.14, -0.17, -0.04, -0.44, -0.03, -0.01, 0.28, -0.38, -0.39, 0.15, 0.53, -0.06, 0.2, -0.18, 0.2, 0.12, 0.13, 0.1, 0.53, 0.04, 0.19, -0.81, 0.3, -0.06, 0.16, -0.2, -0.57, 0.69, -0.04, 0.14 }, {-0.19, 0.1, 0.37, 0.28, 0.16, -0.46, -0.43, -0.09, 0.18, 0.08, 0.14, 0.11, 0.15, 0.48, 0.34, 0.05, 0.04, -0.25, 0.38, 0.61, -0.66, 0.05, 0.1, 0.15, 0.08, 0.05, -0.3, -0.17, 0.29, -0.04, 0.28, -0.29, 0.06, -0.27,

-0.03, 0.16, 0.31, -0.07, 0.16, -0.06, 0.13, -0.02, 0.1, 0.32, 0.15, -0.09, -0.2, 0.17, 0.2, 0.15, -0.1, -0.4, -0.06, -0.16, -0.13, -0.04, -0.18, 0.38, 0.33, -0.18 }, {-0.15, 0.06, 0.49, 0.49, 0.26, -0.23, -0.05, 0.23, 0.18, 0.06, -0.17, 0.01, 0.17, 0.2, 0.01, -0.24, 0.31, -0.53, 0.35, 0.31, -0.63, 0.13, -0.36, 0.31, -0.16, 0.21, -0.32, 0.14, 0.27, -0.8, 0.36, -0.54, -0.03, 0.45, -0.04, 0.29, 0.11, -0.22, 0.06, -0.04, -0.46, 0.08, 0.16, 0.22, -0.06, 0.24, -0.12, 0.2, 0.47, 0.16, 0.07, -0.28, 0.2, -0.27, -0.0, 0.33, -0.24, 0.1, 0.5, -0.25 }, {-0.04, 0.15, -0.01, 0.06, 0.2, -0.63, -0.13, -0.1, -0.05, 0.01, 0.11, 0.06, 0.04, 0.47, 0.12, -0.34, 0.27, -0.2, 0.5, -0.31, -0.38, 0.01, -0.17, 0.01, 0.17, 0.33, -0.52, 0.2, 0.1, -0.3, 0.62, -0.31, 0.12, 0.19, 0.03, 0.19, -0.06, -0.5, 0.59, -0.17, 0.22, -0.03, -0.12, 0.37, 0.0, -0.16, -0.36, 0.19, 0.51, 0.11, -0.01, 0.14, 0.06, -0.44, -0.11, 0.26, -0.34, 0.39, 0.19, 0.22 }, {-0.18, 0.16, 0.13, 0.29, 0.05, -0.25, -0.2, -0.13, 0.04, 0.11, 0.14, 0.13, 0.02, 0.67, -0.07, -0.08, 0.39, 0.11, 0.31, -0.56, -0.47, 0.02, -0.24, 0.27, 0.25, 0.29, -0.38, -0.02, -0.21, -0.23, 0.32, -0.47, 0.32, 0.6, -0.12, 0.37, -0.02, -0.13, 0.36, -0.06, -0.29, 0.27, 0.06, 0.6, -0.2, -0.04, -0.35, -0.03, 0.34, 0.11, 0.02, 0.78, 0.22, -0.42, 0.18, 0.28, -0.23, -0.01, 0.43, -0.25 }, {-0.22, 0.3, -0.06, 0.63, 0.29, -0.39, 0.16, 0.21, -0.16, -0.09, 0.11, 0.06, -0.16, 0.59, -0.2, 0.66, 0.31, 0.21, 0.36, 0.19, -0.44, -0.43, -0.39, -0.14, 0.14, 0.52, 0.02, -0.1, 0.03, 0.3, 0.04, -0.31, 0.11, 0.39, 0.31, 0.16, -0.2, 0.05, 0.25, -0.29, 0.16, 0.1, -0.16, 0.1, -0.15, -0.08, -0.27, 0.12, 0.29, -0.16, -0.82, 0.21, 0.07, -0.87, -0.6, 0.62, 0.06, 0.39, 0.32, -0.68 }, {-0.04, 0.09, -0.09, 0.4, 0.43, -0.31, -0.07, 0.55, -0.15, -0.03, -0.1, 0.06, -0.17, 0.42, -0.56, 0.14, 0.48, 0.0, 0.12, -0.22, -0.51, -0.54, 0.01, -0.34, 0.16, 0.6, -0.05, -0.17, 0.08, 0.3, 0.16, -0.68, -0.42, -0.04, 0.27, 0.11, -0.16, 0.05, 0.17, 0.07, 0.53, 0.15, -0.06, 0.14, -0.2, -0.29, -0.07, 0.07, 0.51, -0.17, -0.59, 0.01, 0.06, -1.09, -0.55, 0.55, -0.03, 0.35, 0.27, -0.75 }, {0.24, -0.55, 0.51, 0.37, -0.11, -0.11, -0.39, 0.13, 0.11, 0.11, -0.31, 0.12, 0.09, 0.49, -0.55, 0.04, 0.49, 0.19, -0.02, -0.8, 0.22, 0.26, 0.69, -0.19, -0.41, 0.05, 0.08, -0.42, -0.84, -0.36, -0.42, -0.46, -0.11, 0.09, -0.09, -0.36, 0.15, -0.18, 0.08, -0.05, 0.39, 0.08, 0.25, -0.08, -0.23,

-0.07, -0.16, -0.22, 0.03, 0.16, -0.07, 0.19, -0.17, 0.0, 0.01, 0.06, -0.22, -0.25, 0.21, -0.4 }, {0.2, -0.66, 0.53, 0.38, -0.07, -0.1, -0.43, 0.23, 0.11, 0.09, -0.33, 0.09, 0.07, 0.46, -0.64, 0.09, 0.5, 0.23, -0.0, -0.78, 0.27, 0.21, 0.72, -0.16, -0.34, 0.12, 0.05, -0.5, -0.89, -0.33, -0.36, -0.43, -0.11, 0.04, -0.08, -0.48, 0.17, -0.21, 0.13, -0.13, 0.39, -0.03, 0.37, -0.18, -0.22, -0.01, -0.08, -0.17, -0.13, 0.16, -0.03, 0.18, -0.12, -0.04, 0.14, 0.03, -0.22, -0.26, 0.15, -0.44 }, {0.18, -0.39, 0.59, 0.56, -0.3, 0.21, 0.02, 0.18, 0.06, -0.0, -0.26, -0.01, 0.04, 0.88, -0.42, 0.29, 0.33, 0.07, 0.28, -0.36, 0.08, 0.2, 0.35, 0.09, -0.72, 0.21, -0.01, -0.35, -0.56, -0.41, -0.49, -0.49, 0.08, 0.44, -0.12, -0.43, 0.17, 0.02, -0.25, -0.02, 0.13, -0.15, 0.36, -0.25, -0.13, -0.0, -0.08, -0.19, -0.02, 0.06, -0.29, 0.75, -0.22, -0.12, -0.01, 0.32, -0.34, -0.01, 0.14, -0.68 }, {0.66, -0.04, 0.53, -0.12, -0.41, 0.47, -0.08, -0.32, 0.14, -0.0, 0.04, -0.0, 0.16, 0.57, -0.3, 0.16, 0.33, 0.38, 0.45, -0.81, -0.06, 0.06, 0.42, 0.01, -0.74, 0.19, -0.06, -0.11, -0.42, -0.78, -0.47, -0.24, 0.26, 0.26, -0.25, -0.62, 0.18, -0.26, -0.07, 0.17, 0.2, -0.24, 0.45, -0.25, -0.03, 0.01, -0.1, -0.11, -0.1, 0.12, 0.18, 0.65, -0.52, 0.14, 0.02, 0.17, 0.14, -0.19, -0.04, -0.35 }, {0.21, -0.22, 0.53, -0.17, 0.41, -0.27, -0.26, 0.37, 0.19, 0.03, -0.51, 0.1, 0.11, 0.38, -0.01, -0.09, 0.45, 0.01, 0.33, -0.14, 0.39, -0.05, 0.4, -0.32, -0.55, 0.1, 0.37, 0.37, -0.34, -0.13, -0.16, -0.59, 0.53, 0.25, -0.25, -0.39, 0.19, -0.04, 0.5, -0.13, -0.23, -0.24, -0.28, -0.18, -0.2, -0.16, -0.28, -0.4, 0.24, 0.17, 0.06, 0.44, -0.17, -0.24, 0.04, 0.03, -0.29, -0.13, 0.11, -0.19 }, {0.13, -0.38, 0.3, -0.16, -0.03, 0.12, -0.4, 0.04, 0.13, 0.12, -0.53, 0.04, 0.05, 0.57, -0.11, 0.28, 0.24, 0.08, 0.12, 0.07, 0.52, 0.05, 0.17, -0.24, -0.38, 0.1, 0.65, 0.31, -0.39, 0.24, -0.44, 0.11, 0.53, 0.3, -0.29, -0.24, 0.11, 0.08, 0.34, -0.35, -0.51, -0.17, -0.4, -0.13, -0.38, -0.32, -0.34, -0.41, -0.14, 0.16, 0.05, 0.05, -0.59, -0.04, -0.04, 0.15, 0.43, -0.51, 0.01, 0.14 }, {-0.25, -0.55, 0.01, -0.05, 0.29, 0.06, -0.51, -0.16, 0.27, 0.04, -0.77, 0.04, 0.14, -0.0, 0.21, -0.07, 0.2, -0.07, 0.12, 0.34, 0.51, -0.06, 0.22, 0.18, 0.05, 0.2, 0.33, 0.18, -0.05, 0.43, -0.32, -0.66, 0.23, -0.14, 0.0, 0.28, 0.15, 0.27, 0.43, 0.29, -0.27, -0.12, -0.34, -0.05, -0.45, -0.23, -0.17, -0.01, -0.06, 0.13, -0.06, -0.25, -0.07, 0.46,

0.12, 0.19, 0.24, -0.02, -0.23, 0.11 }, {-0.5, -0.71, 0.14, 0.0, 0.51, -0.11, -0.34, -0.28, 0.26, 0.13, -0.76, 0.07, 0.16, -0.2, 0.23, -0.52, -0.06, -0.19, -0.05, 0.31, 0.54, 0.08, 0.78, 0.22, -0.1, 0.04, 0.06, 0.07, 0.42, -0.8, -0.12, -0.78, 0.27, -0.51, -0.02, 0.62, 0.12, 0.2, 0.07, -0.27, -0.39, -0.32, -0.53, -0.01, 0.12, -0.44, -0.02, -0.19, -0.14, 0.05, 0.01, 0.08, 0.22, 0.61, -0.02, -0.39, 0.07, 0.24, -0.28, 0.06 }, {-0.34, -0.34, 0.15, -0.19, 0.5, -0.47, -0.46, 0.09, -0.0, 0.06, -0.41, 0.13, 0.11, 0.33, 0.16, -0.49, 0.11, 0.32, 0.03, -0.13, 0.39, 0.45, 0.6, 0.18, -0.18, -0.12, 0.13, 0.03, 0.24, -1.15, -0.24, -0.77, 0.17, -0.78, -0.39, 0.39, 0.1, -0.23, 0.56, -0.17, 0.07, -0.17, -0.02, -0.37, 0.31, -0.27, -0.35, -0.54, 0.21, 0.06, 0.59, 0.15, 0.16, 0.58, 0.33, -0.45, -0.66, 0.25, -0.43, 0.3 }, {0.08, -0.1, -0.09, -0.18, 0.15, -0.52, -0.26, 0.26, -0.03, 0.08, -0.23, 0.06, 0.11, 0.37, -0.1, -0.01, 0.31, 0.24, 0.28, 0.07, 0.16, -0.19, -0.09, -0.14, -0.06, -0.17, -0.24, -0.15, 0.47, -0.17, -0.43, -0.24, 0.06, -0.53, -0.02, 0.16, -0.0, -0.27, 0.12, 0.01, 0.42, -0.28, 0.13, -0.65, 0.35, -0.31, -0.37, -0.28, 0.11, 0.08, 0.35, -0.23, -0.1, 0.43, 0.13, -0.15, -0.7, 0.27, -0.46, 0.08 }, {-0.44, 0.49, -0.07, 0.09, 0.57, -0.75, -0.28, -0.25, 0.06, 0.07, -0.05, 0.09, 0.03, 0.52, 0.1, 0.55, 0.31, 0.31, 0.12, 0.04, 0.09, -0.4, 0.28, -0.18, 0.16, 0.24, -0.14, -0.01, 0.14, -0.12, -0.14, -0.45, 0.16, -0.29, 0.25, 0.42, 0.07, 0.28, 0.43, 0.06, 0.09, 0.04, 0.05, 0.07, -0.13, 0.13, -0.5, 0.1, -0.14, 0.07, -0.17, 0.25, -0.04, -0.02, -0.05, 0.05, -0.25, 0.09, 0.1, -0.19 }, {-0.51, 0.26, -0.24, 0.37, 0.55, 0.08, 0.21, 0.23, 0.02, 0.08, -0.14, -0.0, 0.12, 0.67, -0.23, -0.45, 0.2, -0.27, 0.34, 0.19, -0.27, -0.04, -0.49, 0.31, 0.16, 0.23, -0.09, 0.17, 0.43, -0.15, -0.16, -0.16, -0.09, -0.05, 0.3, 0.2, -0.15, -0.29, -0.0, 0.12, 0.19, 0.01, -0.16, 0.15, -0.26, 0.12, -0.01, 0.31, 0.38, 0.0, -0.24, -0.14, -0.09, -0.28, 0.12, -0.16, -0.49, 0.42, 0.16, 0.07 }, {-0.19, 0.41, -0.17, 0.25, 0.32, -0.73, 0.07, -0.13, -0.12, 0.03, -0.03, 0.12, 0.0, 0.94, -0.05, -0.19, 0.11, -0.13, 0.7, 0.22, -0.1, -0.22, -0.53, -0.28, 0.27, 0.29, -0.37, 0.06, 0.57, 0.15, 0.2, -0.34, 0.0, 0.35, 0.21, -0.03, -0.13, -0.57, 0.37, 0.03, 0.19, -0.13, 0.08, 0.25, -0.01, -0.17, -0.35, 0.23, 0.51, 0.05, -0.21, -0.19, -0.31, -0.71, -0.14, 0.21, -0.12, 0.5, 0.25, 0.04 }, {-0.04, 0.01,

0.1, 0.64, 0.01, -0.26, 0.09, -0.09, -0.05, -0.02, -0.11, 0.03, 0.04, 1.1, -0.45, 0.16, 0.31, 0.01, 0.34, 0.24, -0.6, -0.34, -0.58, 0.03, 0.38, 0.37, -0.3, -0.19, 0.16, 0.04, 0.41, -0.83, 0.12, 0.85, 0.2, -0.02, -0.0, -0.39, 0.05, 0.04, -0.11, -0.01, 0.28, 0.44, -0.18, -0.03, -0.25, 0.16, 0.44, 0.0, -0.62, -0.01, -0.2, -0.8, -0.22, 0.43, -0.31, 0.22, 0.62, -0.55 }, {0.05, 0.21, 0.02, 0.48, 0.29, -0.42, 0.26, 0.24, -0.08, -0.02, 0.08, -0.01, -0.07, 0.79, -0.42, 0.6, 0.18, -0.09, 0.46, -0.03, -0.36, -0.37, -0.5, 0.17, 0.06, 0.83, -0.01, -0.16, -0.28, -0.04, 0.04, -0.63, -0.01, 0.36, 0.34, -0.29, -0.21, 0.24, 0.39, -0.17, -0.11, 0.13, -0.19, -0.1, -0.28, -0.14, -0.17, 0.2, 0.59, -0.11, -0.79, -0.15, -0.21, -1.2, -0.7, 0.73, -0.16, 0.42, 0.26, -0.7 }, {-0.24, -0.03, 0.02, 0.44, 0.57, -0.02, 0.29, 0.31, -0.05, -0.05, -0.34, -0.02, -0.12, 0.71, -0.71, 0.38, 0.2, -0.38, 0.1, -0.21, -0.62, -0.74, -0.03, -0.22, 0.47, 0.87, -0.12, -0.15, 0.21, 0.13, 0.26, -0.68, -0.25, -0.04, 0.46, 0.08, -0.13, 0.13, 0.45, -0.0, 0.65, -0.01, -0.28, 0.2, -0.39, -0.3, 0.07, 0.14, 0.5, -0.09, -0.84, -0.5, -0.3, -1.46, -0.9, 0.8, -0.04, 0.53, 0.37, -0.7 }, {0.11, -0.67, 0.56, 0.45, 0.05, 0.03, -0.68, 0.26, 0.19, 0.03, -0.39, 0.06, 0.1, 0.36, -0.82, -0.15, 0.63, 0.09, -0.19, -0.27, 0.17, 0.47, 0.76, -0.21, -0.37, -0.08, 0.13, -0.49, -0.95, -0.39, -0.08, -0.79, 0.11, -0.14, -0.56, -0.24, 0.18, -0.44, -0.06, 0.03, 0.14, 0.01, 0.26, -0.03, -0.18, -0.06, -0.12, -0.53, -0.16, 0.12, 0.0, 0.22, -0.13, 0.34, 0.06, -0.28, -0.41, -0.3, 0.11, -0.27 }, {0.12, -0.69, 0.57, 0.48, 0.01, -0.01, -0.62, 0.35, 0.1, 0.1, -0.35, 0.05, 0.16, 0.33, -0.74, -0.0, 0.66, 0.19, -0.16, -0.3, 0.27, 0.52, 0.86, -0.25, -0.33, -0.04, 0.11, -0.52, -0.97, -0.42, -0.05, -0.72, 0.06, -0.12, -0.55, -0.14, 0.21, -0.43, -0.08, 0.07, 0.14, -0.03, 0.3, -0.01, -0.16, -0.05, -0.11, -0.61, -0.25, 0.11, 0.07, 0.15, -0.13, 0.32, 0.12, -0.2, -0.5, -0.36, 0.08, -0.35 }, {0.15, -0.61, 0.61, 0.99, -0.4, 0.47, 0.03, 0.01, 0.11, 0.02, -0.39, 0.02, 0.12, 0.52, -0.27, 0.23, 0.45, -0.1, 0.03, -0.4, -0.02, 0.26, 0.78, 0.1, -0.8, -0.13, 0.09, -0.17, -0.7, -0.48, -0.48, -0.63, 0.5, 0.52, -0.32, -0.32, 0.14, -0.24, -0.43, -0.23, -0.0, -0.15, 0.34, -0.18, -0.11, 0.04, 0.07, -0.29, -0.14, 0.09, -0.31, 0.69, -0.31, 0.28, -0.02, 0.15, -0.38, -0.04, 0.08, -0.56 }, {0.72, -0.15, 0.52, 0.44, -0.44, 0.68, 0.24, -0.31, 0.16, 0.04, -0.15,

0.05, 0.14, 0.27, -0.14, 0.22, 0.21, 0.29, 0.4, -0.84, -0.32, 0.13, 0.55, 0.17, -0.86, -0.08, 0.24, -0.01, -0.48, -0.79, -0.61, -0.32, 0.32, 0.52, -0.21, -0.79, 0.13, -0.6, -0.57, -0.02, 0.02, -0.13, 0.37, -0.46, -0.02, 0.02, 0.02, -0.19, -0.12, 0.11, -0.15, 0.59, -0.44, 0.5, 0.14, 0.02, -0.31, -0.41, -0.1, -0.39 }, {0.31, -0.21, 0.51, -0.02, 0.07, 0.25, -0.04, 0.64, 0.11, 0.03, -0.7, 0.11, 0.16, 0.41, -0.37, -0.3, 0.52, -0.37, 0.18, 0.12, 0.27, 0.38, 0.21, 0.08, -0.19, 0.21, 0.36, 0.24, -0.35, -0.41, -0.4, -0.59, 0.02, 0.14, -0.26, -0.37, 0.25, -0.09, -0.08, 0.21, -0.3, -0.28, -0.02, -0.32, -0.28, 0.15, -0.12, -0.09, 0.33, 0.19, -0.16, 0.36, -0.16, 0.06, 0.38, -0.1, -0.32, -0.35, -0.03, -0.24 }, {0.2, -0.3, 0.26, -0.11, 0.03, 0.26, -0.19, 0.09, 0.05, 0.09, -0.62, 0.12, 0.02, 0.26, -0.24, 0.03, 0.23, -0.17, 0.22, 0.06, 0.22, 0.11, 0.3, -0.12, 0.02, 0.36, 0.47, 0.39, -0.16, -0.28, -0.34, 0.02, 0.13, 0.09, -0.12, -0.13, 0.14, 0.29, 0.1, -0.03, -0.44, -0.17, -0.08, 0.21, -0.46, -0.1, -0.27, -0.02, 0.34, 0.13, -0.13, 0.21, -0.35, 0.15, 0.16, 0.08, 0.48, -0.1, -0.01, 0.08 }, {0.33, -0.08, 0.0, -0.26, 0.13, 0.09, -0.05, 0.45, -0.09, 0.08, -0.47, 0.05, 0.0, -0.13, 0.37, -0.76, 0.22, -0.33, 0.44, -0.19, 0.34, 0.08, 0.29, 0.44, -0.28, 0.54, 0.14, 0.56, 0.01, -0.16, -0.16, -0.13, -0.11, -0.25, -0.2, 0.21, 0.07, 0.26, 0.24, -0.4, -0.08, -0.0, -0.27, -0.02, -0.16, -0.43, -0.51, -0.03, 0.85, 0.06, 0.4, 0.17, 0.19, 0.05, 0.19, 0.71, 0.16, 0.81, -0.32, 0.2 }, {-0.24, -0.55, 0.18, -0.24, 0.2, 0.08, -0.02, -0.37, -0.01, 0.03, -0.47, 0.1, 0.06, -0.17, -0.07, -0.6, -0.27, -0.46, -0.18, -0.67, 0.24, 0.29, 0.64, 0.46, -0.11, 0.46, -0.02, 0.49, 0.09, -0.85, 0.63, -0.35, 0.01, -0.03, -0.14, 0.47, 0.1, 0.45, 0.2, 0.24, -0.27, -0.0, -0.13, 0.44, -0.44, -0.28, -0.2, 0.18, -0.02, 0.09, -0.04, 0.3, 0.34, 0.01, -0.05, -0.36, 0.02, 0.68, -0.03, -0.39 }, {-0.02, -0.16, 0.06, -0.55, 0.01, 0.2, -0.21, -0.3, -0.04, 0.14, -0.44, 0.1, 0.08, -0.21, 0.3, -0.71, -0.29, -0.18, -0.35, -0.21, 0.58, 0.64, 0.33, 0.52, 0.1, 0.14, 0.34, 0.22, -0.09, -0.84, -0.15, -0.07, 0.02, -0.54, -0.31, 0.14, 0.04, -0.13, 0.37, 0.23, 0.24, 0.07, -0.06, -0.05, 0.12, -0.16, 0.11, -0.43, -0.1, 0.03, 0.47, 0.01, -0.09, 0.14, 0.45, -0.56, -0.22, 0.45, -0.25, 0.28 }, {0.23, -0.1, -0.1, -0.44, -0.26, 0.3, -0.12, -0.27, -0.01, 0.11, -0.19, -0.0, 0.11, -0.02, 0.04, -0.44, -0.1, -0.08, -0.26,

-0.06, 0.29, 0.2, -0.09, 0.3, 0.28, -0.1, 0.09, 0.09, 0.16, -0.18, -0.24, 0.15, 0.02, -0.36, -0.15, 0.05, -0.02, -0.22, -0.14, 0.24, 0.34, -0.0, 0.13, -0.16, 0.13, -0.13, -0.11, -0.21, -0.22, 0.07, 0.28, -0.33, -0.23, 0.14, 0.34, -0.46, -0.32, 0.04, -0.25, 0.18 }, {-0.26, 0.33, -0.07, -0.03, 0.16, -0.25, -0.16, -0.11, -0.1, 0.04, -0.22, 0.08, 0.03, 0.06, 0.16, -0.2, -0.08, 0.14, 0.11, -0.06, 0.33, -0.09, 0.02, 0.13, 0.08, -0.09, 0.03, 0.09, 0.1, 0.02, -0.09, 0.05, 0.09, -0.2, 0.16, 0.21, -0.02, -0.01, -0.13, -0.08, 0.17, 0.1, 0.04, 0.06, 0.08, -0.03, -0.34, 0.02, 0.1, 0.02, -0.01, 0.01, -0.17, 0.02, 0.1, 0.09, -0.11, 0.21, -0.04, -0.02 }, {-0.57, 0.13, -0.23, 0.24, 0.17, 0.2, 0.0, 0.09, 0.08, 0.04, -0.09, 0.05, 0.1, -0.14, -0.1, -0.13, -0.02, -0.16, -0.01, 0.41, 0.11, -0.19, -0.22, 0.13, 0.25, -0.4, 0.17, -0.23, 0.58, 0.31, 0.11, 0.2, 0.12, -0.46, 0.3, -0.07, -0.03, -0.07, -0.38, 0.55, 0.14, 0.08, 0.19, 0.27, -0.27, 0.41, 0.27, 0.33, -0.07, 0.11, -0.6, -0.03, -0.35, 0.12, 0.02, -0.49, -0.31, 0.21, -0.01, -0.02 }, {-0.15, 0.34, -0.09, -0.14, 0.18, -0.28, -0.1, -0.25, 0.09, 0.05, -0.03, 0.02, -0.02, 0.44, -0.1, 0.05, 0.0, -0.24, 0.33, 0.26, 0.54, -0.04, -0.08, -0.23, 0.37, 0.05, 0.07, -0.27, 0.41, 0.01, 0.07, -0.13, 0.11, -0.17, -0.01, -0.03, -0.12, -0.15, 0.21, 0.24, -0.31, -0.07, 0.28, 0.18, -0.16, 0.0, -0.21, 0.27, 0.1, 0.02, -0.25, 0.11, -0.33, -0.3, -0.03, -0.14, 0.22, 0.08, -0.07, -0.07 }, {-0.1, 0.02, 0.04, 0.31, 0.23, -0.31, 0.0, -0.28, -0.07, 0.08, 0.2, 0.01, 0.07, 0.58, -0.08, 0.28, -0.02, 0.09, 0.24, 0.24, 0.14, -0.44, -0.25, -0.14, 0.07, 0.03, 0.15, -0.27, 0.34, -0.01, -0.02, 0.05, -0.06, 0.07, 0.2, 0.09, -0.04, 0.15, 0.25, -0.38, -0.55, -0.07, 0.07, 0.37, 0.12, -0.04, -0.14, 0.12, 0.11, -0.01, -0.39, 0.09, -0.23, -0.51, -0.17, 0.81, 0.39, 0.17, 0.18, -0.19 }, {-0.23, 0.15, -0.11, 0.12, 0.3, -0.32, 0.1, -0.03, -0.25, -0.08, 0.33, -0.01, -0.12, 0.21, -0.31, 0.01, 0.04, 0.09, 0.34, -0.16, -0.18, -0.07, -0.38, -0.0, 0.01, 0.37, -0.13, -0.21, -0.04, -0.02, 0.65, 0.12, -0.11, 0.12, 0.05, 0.18, -0.16, 0.33, 0.31, -0.31, -0.37, 0.07, -0.12, 0.45, -0.04, -0.25, -0.24, 0.11, 0.19, -0.04, -0.22, -0.09, -0.05, -0.61, -0.42, 0.53, -0.09, 0.38, 0.15, -0.09 }, {-0.29, -0.37, -0.07, 0.4, 0.26, 0.09, 0.1, -0.21, -0.02, -0.0, 0.04, 0.12, -0.04, 0.51, -0.76, 0.27, 0.01, -0.09, -0.16, -0.43, -0.38, -0.39, 0.22, -0.11, 0.42,

0.45, -0.29, -0.32, -0.12, -0.28, 1.09, -0.18, 0.14, 0.12, 0.09, 0.3, -0.09, 0.37, 0.16, -0.11, 0.06, 0.11, 0.03, 0.55, -0.1, -0.08, -0.15, 0.01, -0.25, -0.01, -0.5, -0.23, 0.21, -0.95, -0.51, 0.5, 0.24, -0.13, 0.39, -0.44 }, {-0.28, -0.55, 0.36, 0.28, 0.15, 0.23, -0.04, 0.05, 0.09, 0.1, -0.22, 0.03, 0.18, -0.21, 0.08, -0.15, 0.15, -0.17, -0.38, 0.07, 0.07, 0.44, 0.28, 0.3, -0.2, -0.48, 0.11, -0.37, -0.31, -0.71, -0.13, -0.3, 0.04, 0.12, -0.4, 0.07, 0.22, -0.27, -0.36, -0.1, -0.24, -0.2, 0.08, -0.1, -0.01, -0.23, 0.18, -0.11, -0.42, 0.11, -0.2, 0.24, 0.0, 0.45, 0.1, -0.38, -0.27, -0.17, 0.07, -0.0 }, {-0.22, -0.69, 0.38, 0.3, 0.11, 0.18, -0.13, 0.03, 0.15, 0.1, -0.29, 0.09, 0.11, -0.21, 0.0, -0.11, 0.21, -0.25, -0.48, 0.09, 0.2, 0.46, 0.36, 0.29, -0.18, -0.44, 0.09, -0.47, -0.31, -0.66, 0.1, -0.33, 0.06, 0.13, -0.43, 0.12, 0.13, -0.08, -0.31, -0.01, -0.28, -0.22, 0.13, 0.09, -0.02, -0.27, 0.1, -0.21, -0.52, 0.16, -0.15, 0.33, 0.01, 0.42, 0.1, -0.36, -0.37, -0.2, 0.02, -0.07 }, {-0.18, -0.44, 0.36, 0.67, -0.1, 0.55, 0.27, 0.18, 0.04, 0.03, -0.38, 0.1, 0.1, 0.35, -0.18, -0.17, 0.18, -0.24, -0.01, 0.06, 0.11, 0.23, 0.27, 0.28, -0.45, -0.4, 0.14, -0.21, -0.17, -0.15, -0.39, -0.23, 0.16, 0.2, -0.21, -0.12, 0.13, -0.36, -0.57, -0.3, -0.23, -0.2, -0.01, -0.32, 0.05, -0.22, 0.01, -0.11, 0.03, 0.09, -0.08, 0.38, -0.29, 0.41, 0.06, -0.02, -0.6, 0.11, -0.25, -0.03 }, {0.02, -0.03, 0.24, 0.54, 0.07, 0.23, 0.21, -0.29, 0.12, 0.01, -0.27, 0.06, 0.04, 0.21, 0.11, -0.03, 0.24, 0.27, 0.38, -0.55, -0.11, 0.06, 0.24, 0.14, -0.21, -0.12, 0.28, -0.01, -0.12, -0.24, -0.35, 0.01, 0.05, 0.37, -0.09, -0.3, 0.15, -0.43, -0.37, -0.42, -0.28, -0.1, -0.06, -0.43, 0.19, -0.17, -0.22, -0.19, 0.02, 0.14, 0.03, 0.45, -0.49, 0.37, 0.05, 0.23, -0.12, -0.3, -0.29, -0.05 }, {0.14, -0.45, 0.42, 0.26, -0.21, -0.1, -0.14, 0.64, 0.17, 0.07, -0.57, 0.12, 0.04, 0.36, -0.27, 0.07, 0.62, -0.47, -0.08, 0.0, 0.62, 0.27, 0.51, -0.05, 0.0, -0.14, 0.34, -0.05, -0.42, -0.59, -0.09, -0.63, -0.13, 0.15, -0.34, -0.21, 0.2, -0.0, 0.1, 0.02, -0.55, -0.2, 0.05, 0.0, -0.08, 0.23, -0.04, 0.04, 0.03, 0.1, 0.02, 0.24, 0.0, 0.31, 0.36, -0.04, 0.14, -0.69, -0.04, -0.55 }, {0.04, -0.19, 0.27, 0.04, 0.49, -0.42, -0.4, 0.43, 0.03, 0.01, -0.48, 0.12, 0.03, 0.74, -0.27, 0.21, 0.71, -0.16, 0.3, -0.02, 0.6, 0.14, 0.52, -0.45, 0.06, 0.23, 0.47, 0.19, -0.41, 0.17, -0.11, -0.54, 0.12, -0.04, -0.38, -0.18,

0.18, 0.38, 0.65, -0.13, -0.48, -0.19, -0.27, 0.04, -0.11, -0.13, -0.62, -0.16, 0.26, 0.1, 0.06, 0.13, -0.34, 0.05, 0.04, 0.43, -0.09, -0.3, -0.04, -0.32 }, {0.25, -0.11, 0.17, -0.24, 0.53, 0.03, -0.35, 0.73, 0.07, 0.03, -0.52, 0.13, 0.13, 0.35, -0.17, -0.59, 0.61, -0.58, 0.38, 0.15, 0.68, 0.33, 0.3, 0.18, 0.2, 0.3, 0.41, 0.16, -0.3, 0.38, -0.29, -0.21, 0.13, -0.64, -0.5, 0.11, 0.17, 0.22, 0.45, 0.14, -0.5, 0.13, -0.41, -0.22, 0.02, -0.19, -0.41, -0.24, 0.55, 0.1, 0.32, -0.13, -0.26, 0.11, 0.48, 0.18, -0.16, 0.31, -0.53, 0.25 }, {-0.33, -0.22, 0.34, -0.17, 0.84, -0.2, -0.15, 0.38, 0.14, 0.09, -0.5, 0.13, 0.12, 0.31, -0.44, -0.23, 0.22, -0.41, -0.07, 0.05, 0.15, 0.26, 0.46, 0.06, 0.38, 0.67, 0.22, 0.17, 0.17, -0.09, 0.34, -0.19, 0.22, -0.35, -0.13, 0.26, 0.1, 0.28, 0.78, 0.3, -0.09, -0.0, -0.21, 0.14, -0.04, -0.01, -0.17, -0.07, 0.37, 0.11, 0.08, -0.39, -0.13, 0.08, 0.21, -0.2, -0.19, 0.54, -0.3, -0.11 }, {-0.12, 0.04, 0.12, -0.36, 0.17, -0.4, 0.09, -0.3, -0.07, 0.04, -0.56, 0.02, 0.06, -0.18, 0.29, -0.37, -0.23, 0.06, -0.21, -0.1, 0.38, 0.42, 0.66, 0.16, 0.07, 0.46, 0.04, 0.25, -0.08, -0.06, -0.06, 0.04, 0.15, -0.42, -0.17, 0.07, -0.11, -0.16, 0.25, 0.15, 0.45, 0.02, -0.09, -0.15, 0.07, 0.09, 0.17, -0.46, 0.02, 0.07, 0.56, -0.35, 0.05, -0.17, 0.36, -0.31, -0.23, 0.43, -0.3, 0.5 }, {-0.15, 0.03, 0.03, -0.23, -0.07, -0.06, 0.17, -0.32, -0.07, 0.05, -0.31, 0.02, 0.03, -0.4, 0.26, -0.06, -0.52, 0.02, -0.19, 0.15, -0.05, 0.13, 0.21, 0.19, 0.29, 0.1, -0.02, 0.13, 0.2, 0.15, -0.01, 0.24, 0.16, -0.26, -0.01, 0.15, -0.02, -0.36, -0.07, 0.08, 0.56, -0.01, -0.04, -0.14, 0.19, 0.06, 0.23, -0.3, -0.26, -0.01, 0.26, -0.51, -0.17, 0.1, 0.26, -0.23, 0.14, 0.24, -0.26, 0.36 }, {-0.08, 0.15, -0.13, -0.21, -0.29, -0.24, 0.14, -0.27, -0.19, 0.08, -0.02, 0.03, 0.01, -0.11, 0.09, -0.11, -0.41, 0.17, 0.14, -0.1, 0.1, 0.06, -0.09, 0.11, 0.05, -0.26, -0.07, -0.07, 0.17, 0.2, -0.21, 0.28, 0.14, -0.09, 0.14, 0.06, -0.0, -0.19, -0.36, 0.09, 0.16, 0.04, 0.04, -0.2, -0.0, 0.23, -0.17, -0.01, 0.08, -0.02, -0.1, 0.18, -0.26, 0.16, 0.22, -0.12, 0.15, 0.2, -0.2, 0.36 }, {-0.33, 0.45, -0.12, -0.15, 0.26, -0.43, -0.09, 0.15, 0.07, 0.05, -0.08, 0.05, 0.13, -0.17, -0.19, -0.27, -0.2, -0.09, 0.46, -0.02, -0.03, -0.19, -0.33, 0.06, 0.24, -0.04, -0.04, -0.29, 0.58, 0.39, 0.04, -0.06, 0.07, -0.45, 0.23, -0.28, 0.01, 0.09, 0.11, 0.48, 0.58, 0.09, -0.01, 0.17, -0.28, 0.43, 0.14, 0.3,

0.43, 0.04, -0.25, 0.04, -0.45, -0.37, 0.14, -0.08, -0.18, 0.51, -0.12, 0.13 }, {-0.47, 0.34, 0.02, -0.1, 0.4, -0.91, -0.19, -0.05, 0.06, 0.09, -0.1, 0.07, -0.0, -0.13, -0.33, -0.13, 0.06, -0.32, 0.13, 0.63, 0.38, 0.05, -0.09, -0.18, 0.47, 0.2, 0.2, -0.29, 0.48, 0.06, 0.07, -0.17, -0.01, -0.29, -0.01, 0.24, 0.01, -0.02, 0.3, 0.2, -0.12, 0.0, 0.14, 0.27, -0.21, 0.1, 0.16, 0.12, 0.13, 0.04, -0.08, 0.08, -0.49, -0.29, 0.06, -0.1, 0.13, 0.04, -0.06, 0.04 }, {-0.06, 0.07, 0.0, 0.29, 0.28, -0.69, -0.13, -0.11, 0.06, 0.1, 0.16, 0.06, 0.09, 0.25, -0.25, -0.06, 0.29, 0.12, 0.16, 0.4, 0.14, -0.13, -0.13, -0.25, 0.16, 0.01, 0.05, -0.44, 0.25, 0.05, 0.23, -0.06, -0.05, -0.06, 0.11, 0.17, 0.07, 0.19, 0.27, -0.36, -0.22, 0.01, -0.02, 0.21, 0.09, 0.1, 0.09, -0.04, 0.02, 0.08, -0.18, -0.02, -0.35, -0.45, -0.04, 0.43, 0.05, -0.12, 0.14, -0.03 }, {-0.08, 0.06, -0.18, 0.0, -0.23, -0.08, 0.12, -0.24, -0.18, 0.01, 0.43, 0.11, -0.04, 0.08, -0.41, 0.19, 0.09, 0.11, 0.07, -0.46, 0.02, 0.11, -0.52, 0.17, 0.2, -0.03, -0.25, -0.43, -0.34, -0.12, 0.75, 0.14, -0.22, 0.25, -0.06, 0.08, -0.12, 0.12, 0.01, -0.23, -0.34, 0.16, 0.16, 0.24, -0.05, 0.0, -0.21, 0.18, -0.04, -0.11, -0.21, -0.08, -0.02, -0.06, -0.4, -0.14, 0.24, -0.2, 0.22, 0.01 }, {-0.3, -0.53, -0.05, 0.49, -0.21, 0.18, 0.31, -0.53, -0.04, -0.09, 0.12, 0.07, -0.1, 0.46, -0.73, 0.35, 0.02, -0.04, -0.32, -0.64, -0.61, -0.11, 0.21, 0.17, 0.28, 0.08, -0.65, -0.6, -0.45, -0.61, 1.21, -0.16, -0.01, 0.49, 0.19, 0.11, -0.1, 0.08, -0.05, -0.09, -0.04, 0.23, 0.26, 0.55, -0.07, -0.09, 0.12, 0.09, -0.54, -0.1, -0.67, -0.1, 0.34, -0.6, -0.55, 0.09, 0.14, -0.29, 0.59, -0.36 }, {-0.3, -0.92, 0.32, 0.65, -0.37, 0.86, 0.07, 0.28, 0.13, 0.09, -0.4, 0.11, 0.09, 0.02, -0.19, 0.26, 0.41, -0.36, -0.82, 0.92, -0.08, 0.21, 0.26, 0.28, -0.03, -0.9, -0.02, -0.52, -0.09, -0.63, 0.05, -0.43, -0.01, 0.31, -0.15, 0.29, 0.19, -0.36, -0.74, 0.22, -0.2, -0.21, 0.34, -0.13, 0.04, -0.32, 0.51, -0.09, -0.67, 0.2, -0.48, -0.01, -0.32, 0.62, 0.12, -0.81, -0.42, -0.71, 0.28, -0.51 }, {-0.26, -1.01, 0.39, 0.74, -0.25, 0.85, 0.07, 0.29, 0.04, 0.08, -0.41, 0.1, 0.08, 0.03, -0.26, 0.27, 0.34, -0.38, -0.74, 0.85, -0.06, 0.2, 0.26, 0.34, -0.07, -0.93, 0.09, -0.53, -0.06, -0.56, 0.07, -0.4, 0.04, 0.35, -0.21, 0.32, 0.18, -0.37, -0.69, 0.23, -0.16, -0.23, 0.37, -0.19, 0.07, -0.33, 0.42, -0.18, -0.67, 0.1, -0.43, -0.08, -0.31, 0.6, 0.07, -0.77,

-0.43, -0.74, 0.29, -0.5 }, {-0.38, -0.62, 0.25, 0.75, -0.2, 0.66, 0.24, -0.13, 0.11, 0.05, -0.43, 0.02, 0.17, 0.24, -0.29, 0.44, 0.28, -0.18, -0.31, 0.52, -0.02, -0.03, -0.04, 0.37, -0.22, -0.58, 0.26, -0.43, -0.06, -0.73, -0.42, -0.34, 0.23, 0.53, -0.04, 0.18, 0.15, -0.38, -0.65, -0.2, -0.37, -0.14, -0.03, -0.34, 0.04, -0.12, 0.11, -0.12, -0.46, 0.17, -0.26, 0.21, -0.45, 0.43, -0.01, -0.22, -0.31, -0.52, 0.11, -0.44 }, {-0.23, -0.03, 0.09, 0.64, 0.29, -0.04, 0.38, -0.53, 0.0, 0.01, 0.08, 0.08, 0.1, 0.59, 0.22, 0.49, 0.36, 0.55, 0.41, -0.53, -0.08, -0.03, 0.17, 0.06, -0.08, -0.11, 0.41, -0.24, -0.21, -0.6, -0.48, -0.01, -0.29, 0.34, -0.32, -0.08, 0.11, 0.03, -0.02, -0.65, -0.59, -0.04, -0.3, -0.32, 0.31, -0.24, -0.5, -0.32, -0.2, 0.08, 0.27, 0.38, -0.58, 0.15, -0.05, 0.34, 0.17, -0.57, -0.18, -0.13 }, {-0.17, -0.45, 0.19, 0.49, 0.04, -0.24, -0.13, 0.4, 0.06, 0.09, -0.19, 0.1, 0.02, 0.28, 0.04, 0.31, 0.89, -0.07, 0.02, -0.12, 0.38, 0.18, 0.44, -0.2, -0.05, -0.24, 0.42, -0.03, -0.42, -0.46, -0.25, -0.57, -0.23, 0.21, -0.41, 0.21, 0.24, 0.18, 0.09, -0.16, -0.52, -0.21, -0.05, 0.02, 0.18, 0.08, -0.08, -0.14, -0.07, 0.03, 0.13, 0.52, -0.17, 0.72, 0.17, -0.23, 0.47, -0.88, -0.03, -0.4 }, {0.06, -0.35, 0.21, 0.49, 0.23, -0.56, -0.34, 0.43, -0.06, 0.06, 0.12, 0.08, 0.07, 0.69, -0.01, 0.36, 0.9, 0.48, 0.41, -0.06, -0.01, 0.0, 0.6, -0.55, -0.38, -0.06, 0.37, -0.11, -0.55, 0.23, -0.03, -0.51, -0.16, 0.24, -0.25, 0.01, 0.11, -0.0, 0.36, -0.02, -0.31, -0.05, 0.0, -0.03, 0.28, -0.21, -0.4, -0.56, -0.03, 0.03, -0.11, -0.01, -0.04, 0.21, 0.02, -0.08, -0.34, -0.45, 0.15, -0.56 }, {-0.08, -0.1, 0.16, -0.08, 0.58, -0.48, -0.38, 0.35, 0.08, 0.13, -0.13, 0.05, 0.02, 0.3, -0.23, -0.55, 0.6, -0.1, 0.04, 0.42, 0.71, 0.42, 0.52, 0.03, 0.08, 0.3, 0.48, 0.05, -0.29, 0.16, -0.02, -0.47, 0.33, -0.5, -0.58, 0.49, 0.12, -0.0, 0.35, 0.21, -0.45, 0.21, -0.23, 0.01, 0.14, -0.21, -0.26, -0.57, -0.04, 0.03, 0.18, -0.13, -0.11, 0.07, 0.41, 0.14, 0.16, 0.17, -0.36, 0.03 }, {-0.58, 0.02, 0.27, -0.34, 0.71, -0.39, -0.35, -0.37, 0.19, 0.1, -0.2, 0.08, 0.1, 0.16, -0.35, -0.44, 0.03, 0.06, -0.14, 0.32, 0.49, 0.33, 0.59, 0.07, 0.4, 0.84, 0.43, 0.22, 0.27, -0.31, 0.41, -0.28, 0.44, -0.35, -0.37, 0.57, 0.14, 0.16, 0.5, 0.24, -0.33, 0.04, -0.19, 0.36, -0.0, -0.05, -0.24, -0.25, 0.07, -0.0, -0.04, -0.34, -0.14, 0.2, 0.11, -0.19, 0.3, 0.37, -0.22, -0.02 }, {-0.29, 0.35, 0.17, -0.66,

0.3, -0.65, -0.22, -0.55, -0.08, 0.04, -0.57, 0.02, 0.09, -0.28, -0.01, -0.92, -0.16, 0.51, -0.22, -0.18, 0.74, 0.43, 0.95, 0.11, 0.19, 0.57, 0.27, 0.24, -0.12, -0.17, 0.06, -0.2, 0.34, -0.62, -0.48, 0.38, 0.01, -0.2, 0.24, 0.08, 0.08, 0.12, -0.46, 0.07, 0.17, 0.12, -0.03, -0.66, 0.09, 0.08, 0.65, 0.07, -0.01, -0.4, 0.21, 0.02, -0.17, 0.28, -0.31, 0.65 }, {-0.19, 0.35, 0.08, -0.47, 0.14, -0.72, -0.08, -0.48, -0.11, 0.1, -0.38, 0.03, 0.0, -0.21, -0.06, -0.61, -0.16, 0.42, -0.1, -0.15, 0.53, 0.22, 0.6, 0.09, 0.17, 0.41, 0.45, 0.18, 0.02, -0.16, 0.04, -0.26, 0.49, -0.34, -0.34, 0.32, 0.0, -0.24, 0.04, -0.01, -0.13, 0.15, -0.37, 0.07, 0.1, 0.14, -0.16, -0.6, 0.04, 0.1, 0.41, -0.08, 0.0, -0.29, 0.14, 0.14, -0.12, 0.1, -0.28, 0.56 }, {0.04, 0.25, 0.05, -0.12, -0.42, -0.05, -0.01, -0.38, -0.07, 0.01, -0.3, 0.09, 0.12, -0.14, -0.28, -0.43, -0.13, 0.6, 0.14, -0.12, 0.68, 0.39, 0.35, 0.33, 0.05, -0.05, 0.24, -0.08, 0.01, -0.46, -0.43, -0.17, 0.22, -0.09, -0.3, 0.32, 0.06, -0.11, -0.34, 0.07, -0.48, 0.03, -0.21, -0.04, 0.11, 0.21, -0.17, -0.33, 0.07, 0.05, 0.29, 0.48, 0.14, 0.14, 0.09, -0.08, 0.11, -0.09, -0.37, 0.53 }, {0.32, 0.44, 0.02, -0.07, -0.22, 0.04, 0.01, 0.41, 0.11, 0.05, -0.34, 0.08, 0.02, -0.26, -0.62, -0.23, 0.4, 0.17, 0.37, 0.03, 0.22, 0.26, -0.25, 0.19, -0.06, 0.17, 0.21, -0.35, 0.04, 0.28, -0.55, -0.24, 0.01, -0.08, 0.0, -0.21, -0.05, 0.04, -0.07, 0.61, -0.01, 0.07, 0.01, -0.24, -0.26, 0.74, 0.11, 0.02, 0.5, 0.04, -0.11, 0.29, -0.39, -0.11, 0.23, -0.18, -0.13, -0.03, -0.33, 0.26 }, {-0.1, 0.62, 0.1, -0.12, 0.46, -0.34, -0.13, -0.16, 0.03, 0.06, -0.25, 0.04, 0.03, -0.26, -0.49, -0.3, 0.19, 0.01, 0.33, 0.32, 0.53, 0.04, -0.05, -0.04, 0.35, 0.27, 0.4, -0.57, 0.45, 0.39, -0.37, -0.28, -0.1, -0.1, -0.01, 0.34, -0.04, -0.19, 0.06, 0.32, -0.2, -0.03, -0.04, -0.04, -0.1, 0.3, 0.0, 0.02, 0.21, -0.07, 0.17, -0.1, -0.35, -0.02, 0.08, 0.11, -0.03, -0.13, -0.03, 0.16 }, {0.02, -0.02, 0.07, -0.02, 0.06, 0.36, -0.08, -0.28, 0.02, 0.08, -0.13, -0.01, 0.02, -0.1, -0.45, 0.09, 0.14, 0.28, 0.07, 0.27, 0.35, 0.01, 0.18, -0.05, 0.2, 0.04, 0.18, -0.55, 0.08, 0.2, -0.01, -0.24, -0.19, 0.2, 0.06, -0.02, 0.03, 0.12, -0.24, 0.2, -0.17, 0.01, 0.19, 0.12, -0.16, 0.32, 0.08, -0.15, -0.07, 0.01, -0.13, -0.16, -0.37, 0.03, 0.11, 0.11, -0.11, -0.46, 0.14, 0.02 }, {0.22, 0.22, 0.06, 0.1, -0.21, 0.08, 0.02, -0.13, -0.03, -0.05, 0.04, 0.03, -0.08, -0.37,

-0.45, 0.19, 0.26, 0.02, 0.4, -0.66, -0.1, 0.2, -0.33, 0.08, -0.13, 0.02, -0.37, -0.55, -0.62, -0.37, 0.2, -0.12, -0.38, 0.16, -0.07, -0.18, -0.09, -0.03, -0.38, 0.11, -0.58, 0.06, 0.33, -0.12, -0.14, 0.29, -0.15, -0.24, 0.01, 0.01, -0.08, 0.03, 0.02, 0.16, -0.23, -0.08, 0.11, -0.5, 0.29, -0.12 }, {-0.41, -0.63, 0.13, 0.97, -0.27, 0.44, 0.3, -0.58, -0.01, 0.0, -0.28, 0.01, 0.02, 0.22, -0.62, 0.69, -0.1, -0.19, -0.27, -0.46, -0.84, -0.18, 0.33, 0.3, -0.05, 0.22, -0.7, -0.7, -0.44, -0.85, 0.75, -0.36, -0.08, 0.65, 0.34, 0.11, -0.15, -0.29, -0.32, -0.07, -0.33, 0.15, 0.46, 0.18, -0.19, 0.21, 0.2, -0.32, -0.77, -0.06, -0.78, -0.05, 0.68, -0.32, -0.49, 0.38, 0.49, -0.78, 0.64, -0.76 }, {-0.14, -0.67, 0.06, 0.35, -0.52, 0.76, -0.0, 0.54, 0.02, 0.01, -0.24, -0.01, 0.05, -0.5, -0.32, -0.01, -0.02, 0.06, -0.67, 0.85, -0.09, -0.25, 0.06, 0.15, -0.04, -1.03, 0.11, -0.23, -0.12, 0.08, 0.11, -0.23, 0.55, 0.22, -0.18, 0.04, 0.06, -0.75, -0.73, 0.37, 0.22, -0.01, 0.66, -0.26, -0.29, -0.28, 0.36, -0.12, -0.89, 0.02, -0.7, -0.3, -0.59, 0.46, -0.02, -0.93, -0.17, -0.35, 0.38, -0.33 }, {-0.22, -0.63, 0.14, 0.44, -0.5, 0.75, 0.04, 0.44, 0.05, 0.01, -0.3, 0.1, -0.03, -0.46, -0.25, -0.03, -0.07, 0.01, -0.71, 0.85, -0.14, -0.15, 0.07, 0.2, -0.05, -0.98, 0.09, -0.33, -0.18, 0.01, 0.2, -0.3, 0.48, 0.26, -0.25, 0.08, -0.01, -0.85, -0.62, 0.28, 0.23, -0.01, 0.53, -0.31, -0.38, -0.3, 0.45, -0.12, -0.84, -0.02, -0.62, -0.23, -0.51, 0.54, -0.07, -0.9, -0.19, -0.38, 0.35, -0.38 }, {0.1, -0.06, 0.05, 0.19, -0.4, 0.2, 0.01, 0.53, -0.07, 0.05, -0.23, 0.08, -0.05, 0.17, -0.24, -0.04, 0.21, 0.45, -0.01, 0.61, 0.5, 0.05, -0.15, 0.18, -0.07, -0.54, 0.51, -0.06, -0.13, 0.21, -0.24, -0.22, 0.12, 0.18, -0.23, -0.2, 0.01, -0.74, -0.43, 0.22, -0.09, 0.06, -0.05, -0.44, -0.13, -0.13, -0.1, -0.13, -0.24, 0.04, -0.47, -0.22, -0.84, 0.3, -0.02, -0.4, -0.26, 0.01, 0.05, 0.21 }, {-0.01, 0.41, -0.19, 0.34, 0.02, -0.19, 0.4, 0.3, 0.09, 0.09, -0.06, 0.03, 0.12, 0.62, 0.32, 0.11, 0.24, 0.33, 0.48, 0.11, 0.67, 0.05, -0.46, 0.41, -0.12, -0.06, 0.52, -0.26, -0.33, 0.02, -0.59, -0.02, -0.11, 0.03, -0.13, -0.22, 0.06, -0.06, 0.05, -0.3, -0.71, 0.07, -0.8, -0.28, -0.17, -0.07, -0.34, -0.26, -0.04, 0.06, -0.14, 0.05, -0.94, -0.19, -0.25, 0.45, 0.2, -0.1, -0.29, 0.06 }, {0.3, 0.42, 0.26, -0.3, 0.24, -0.58, -0.28, 0.58, 0.07, 0.01, -0.35, 0.03, 0.06, 0.09, 0.06, -0.23, 0.28, -0.17, 0.53, 0.02,

0.55, 0.12, -0.1, 0.02, -0.15, 0.34, 0.31, -0.21, -0.44, -0.27, -0.62, -0.66, 0.06, -0.21, -0.35, -0.43, 0.13, 0.33, 0.45, 0.14, -0.15, -0.15, -0.34, -0.33, -0.45, 0.15, -0.35, 0.25, 0.77, 0.05, 0.09, 0.47, -0.17, -0.2, -0.13, 0.46, 0.24, 0.09, -0.15, 0.05 }, {0.39, 0.36, 0.1, -0.32, 0.14, -0.56, -0.24, 0.98, 0.02, 0.0, -0.07, 0.11, 0.11, -0.01, -0.03, -0.13, 0.33, -0.09, 0.53, 0.42, 0.06, -0.11, -0.08, -0.18, -0.35, 0.08, 0.34, -0.11, -0.32, 0.2, -0.77, -0.18, 0.16, -0.29, -0.11, -0.51, 0.09, 0.09, 0.36, 0.24, 0.26, -0.24, -0.48, -0.64, -0.46, -0.15, -0.36, 0.18, 0.6, 0.08, 0.14, -0.39, -0.04, -0.3, -0.16, 0.52, 0.04, 0.21, -0.11, 0.14 }, {0.26, 0.46, 0.05, -0.21, 0.05, -0.1, -0.65, 0.53, 0.14, 0.05, -0.05, 0.01, 0.01, -0.04, 0.01, -0.27, -0.12, 0.18, 0.26, 0.45, 0.33, -0.04, 0.29, -0.06, -0.0, 0.24, 0.06, -0.05, -0.01, 0.08, -0.34, 0.01, 0.21, -0.42, -0.37, 0.35, 0.12, 0.05, 0.36, 0.04, 0.48, -0.21, -0.24, -0.29, 0.12, -0.2, -0.25, -0.27, 0.18, 0.01, 0.6, -0.43, 0.06, 0.28, 0.3, -0.09, 0.44, 0.14, -0.46, 0.36 }, {-0.07, 0.31, 0.15, -0.18, 0.35, -0.22, -0.34, -0.08, 0.12, 0.1, -0.0, 0.01, 0.01, -0.22, 0.06, -0.06, -0.36, 0.1, -0.01, 0.39, 0.38, 0.17, 0.33, 0.01, -0.08, 0.54, -0.27, 0.11, 0.17, -0.32, -0.08, 0.13, 0.26, -0.32, -0.34, 0.45, 0.13, 0.11, 0.36, -0.25, 0.29, -0.34, -0.25, -0.16, 0.13, -0.07, -0.06, -0.08, 0.18, 0.11, 0.39, -0.23, 0.23, 0.27, 0.02, -0.47, 0.26, 0.25, -0.29, 0.24 }, {-0.43, 0.57, -0.08, -0.58, 0.58, -0.27, -0.03, -0.52, -0.05, 0.14, -0.1, 0.05, 0.14, -0.17, 0.02, -0.55, -0.44, 0.15, 0.08, -0.25, 0.49, 0.38, 0.35, 0.24, -0.03, 0.46, 0.01, 0.2, 0.17, -0.21, -0.01, 0.16, 0.32, -0.41, -0.19, 0.5, 0.01, 0.23, 0.5, -0.48, -0.03, -0.02, -0.49, -0.04, 0.21, -0.02, -0.22, -0.25, 0.43, 0.03, 0.66, 0.34, 0.19, 0.11, 0.07, 0.11, -0.04, -0.08, -0.65, 0.6 }, {0.07, 0.82, -0.0, -0.5, 0.55, -0.52, -0.04, -0.02, -0.0, 0.04, -0.05, 0.05, 0.03, 0.07, 0.03, -0.38, -0.18, 0.09, 0.26, 0.01, 0.51, 0.2, -0.09, 0.06, -0.11, 0.42, 0.1, 0.15, 0.18, 0.1, -0.21, 0.12, 0.28, -0.36, -0.13, 0.28, -0.0, 0.27, 0.51, -0.36, -0.13, -0.07, -0.39, -0.06, 0.13, 0.13, -0.34, -0.28, 0.36, 0.03, 0.64, 0.21, 0.08, -0.07, 0.15, 0.48, -0.26, -0.06, -0.63, 0.78 }, {-0.18, 0.67, -0.13, -0.45, -0.12, -0.04, -0.03, -0.04, -0.02, 0.05, -0.07, 0.03, 0.03, 0.12, -0.23, -0.02, 0.04, 0.45, 0.15, -0.17, 0.38, 0.05, -0.34, 0.06, 0.28, -0.03,

0.07, -0.12, 0.28, 0.1, -0.29, 0.22, 0.2, -0.04, -0.03, 0.18, -0.04, -0.06, 0.18, -0.11, -0.1, -0.08, -0.32, -0.2, -0.03, 0.1, -0.24, 0.01, 0.46, 0.05, 0.24, 0.18, -0.16, 0.25, -0.01, 0.06, -0.09, -0.1, -0.53, 0.71 }, {-0.2, 0.19, -0.03, -0.53, 0.03, 0.13, -0.14, -0.09, 0.2, 0.04, -0.48, 0.02, 0.15, -0.3, -0.11, -0.09, 0.09, 0.01, -0.2, 0.02, 0.47, 0.24, -0.03, -0.01, 0.12, -0.01, 0.46, 0.03, -0.05, 0.23, 0.16, 0.39, 0.04, -0.52, 0.11, 0.09, 0.0, 0.46, 0.04, 0.15, 0.15, -0.02, -0.26, 0.14, -0.24, 0.42, 0.2, -0.17, -0.21, 0.05, 0.53, 0.15, -0.42, 0.06, 0.32, -0.11, 0.4, -0.15, -0.51, 0.63 }, {-0.29, -0.09, 0.15, -0.15, 0.15, -0.13, -0.26, -0.12, 0.06, 0.1, -0.41, 0.1, 0.08, 0.04, -0.16, -0.29, 0.18, -0.11, -0.19, 0.6, 0.57, 0.32, 0.24, -0.18, 0.67, -0.07, 0.6, -0.18, 0.27, 0.36, -0.06, -0.08, 0.2, -0.47, -0.0, 0.5, 0.01, -0.01, 0.16, 0.26, 0.18, -0.04, -0.2, 0.17, -0.09, 0.31, 0.32, -0.17, -0.24, -0.04, 0.18, -0.37, -0.29, 0.06, 0.21, -0.45, 0.19, -0.29, -0.38, 0.09 }, {-0.07, -0.56, 0.16, -0.2, 0.04, 0.18, -0.4, 0.05, 0.1, 0.07, -0.36, 0.09, 0.05, 0.08, -0.42, 0.11, 0.31, 0.0, -0.47, 0.43, 0.44, 0.25, 0.37, -0.21, 0.53, -0.3, 0.58, -0.41, -0.1, 0.39, -0.03, -0.04, 0.22, -0.25, -0.01, 0.13, -0.02, 0.24, -0.06, 0.23, 0.48, -0.06, 0.24, 0.06, -0.08, 0.54, 0.36, -0.31, -0.75, 0.01, -0.09, -0.25, -0.34, 0.17, 0.29, -0.43, -0.1, -0.67, -0.15, 0.02 }, {0.04, 0.08, 0.13, 0.05, 0.22, -0.28, -0.34, 0.28, -0.04, 0.03, -0.12, 0.04, -0.05, 0.13, -0.09, 0.26, 0.29, -0.16, 0.05, -0.1, -0.12, -0.12, -0.23, -0.26, 0.47, -0.2, 0.03, -0.19, 0.04, -0.33, 0.06, 0.03, -0.24, 0.26, -0.07, 0.13, -0.06, 0.05, -0.21, -0.28, 0.1, -0.12, 0.22, -0.34, 0.15, 0.04, -0.22, -0.12, -0.16, 0.02, -0.18, 0.22, 0.01, 0.09, -0.13, -0.03, 0.22, -0.52, 0.01, -0.44 }, {0.04, -0.24, 0.04, 0.7, 0.13, 0.59, 0.04, -0.29, -0.02, -0.02, -0.12, -0.01, 0.03, 0.37, -0.05, 0.39, -0.11, -0.69, -0.23, -0.3, -0.97, -0.02, 0.11, 0.04, 0.15, -0.14, -0.37, -0.12, -0.03, -1.08, -0.07, 0.0, 0.09, 0.42, 0.16, -0.1, -0.16, -0.15, -0.31, -0.46, 0.01, -0.23, 0.23, -0.21, 0.02, -0.13, -0.11, -0.12, -0.4, -0.02, -0.38, 0.3, 0.53, -0.07, -0.19, 0.31, 0.42, -0.51, 0.21, -0.74 }, {-0.19, -0.4, 0.05, -0.0, -0.3, 0.31, -0.12, 0.04, 0.03, 0.04, -0.18, 0.02, -0.03, -0.42, -0.53, -0.02, -0.07, 0.49, -0.44, 0.69, -0.12, -0.23, 0.17, -0.11, -0.11, -0.61, -0.03, -0.13, -0.08, 0.39, 0.36, -0.12, 0.64, -0.03, -0.17,

0.14, 0.01, -0.41, -0.08, 0.19, 0.25, 0.18, 0.53, -0.11, -0.32, -0.22, 0.19, -0.24, -0.53, -0.04, -0.19, -0.53, -0.57, 0.11, -0.23, -0.66, -0.17, -0.16, 0.3, -0.17 }, {-0.22, -0.4, 0.08, -0.13, -0.42, 0.28, -0.1, 0.11, 0.08, 0.05, -0.09, 0.01, -0.04, -0.33, -0.47, -0.05, -0.15, 0.5, -0.33, 0.59, -0.17, -0.22, 0.11, -0.06, 0.03, -0.64, -0.03, -0.3, 0.06, 0.34, 0.47, -0.1, 0.62, -0.06, -0.17, 0.04, -0.08, -0.4, -0.06, 0.23, 0.36, 0.19, 0.58, -0.05, -0.3, -0.29, 0.11, -0.04, -0.42, 0.03, -0.18, -0.52, -0.53, 0.12, -0.24, -0.73, -0.08, -0.15, 0.33, -0.1 }, {-0.1, 0.4, 0.07, -0.02, 0.15, -0.28, -0.03, 0.7, 0.06, 0.05, -0.15, 0.0, -0.04, 0.4, -0.14, -0.24, 0.24, 0.66, 0.25, 0.6, 0.49, 0.05, 0.01, -0.16, -0.26, -0.22, 0.53, 0.04, -0.25, 0.63, -0.25, -0.44, 0.13, -0.28, -0.18, -0.02, 0.04, -0.54, 0.31, 0.04, -0.12, 0.16, -0.38, -0.39, -0.16, -0.22, -0.25, -0.25, 0.05, 0.06, 0.0, -0.48, -0.84, -0.21, -0.11, -0.1, -0.67, 0.04, -0.09, 0.45 }, {-0.06, 0.71, -0.13, 0.2, 0.38, -0.55, 0.13, 0.39, 0.1, 0.02, 0.0, 0.02, 0.1, 0.61, 0.31, -0.05, 0.35, 0.41, 0.67, 0.12, 0.76, 0.12, -0.62, 0.13, -0.13, -0.03, 0.54, -0.3, -0.24, 0.31, -0.69, -0.25, -0.02, -0.14, -0.1, -0.15, -0.03, 0.15, 0.55, -0.15, -0.62, 0.02, -0.68, -0.45, -0.27, -0.06, -0.44, 0.14, 0.27, 0.07, -0.03, -0.21, -0.88, -0.24, -0.28, 0.29, -0.62, 0.14, -0.35, 0.08 }, {-0.19, 0.2, 0.26, 0.33, 0.32, -0.58, -0.41, 0.1, 0.24, 0.05, -0.06, 0.07, 0.11, 0.48, -0.26, 0.22, 0.43, 0.41, 0.24, -0.37, 0.57, 0.31, -0.21, -0.13, 0.26, -0.12, 0.52, -0.21, -0.54, -0.53, -0.01, -0.55, 0.17, 0.07, -0.38, 0.03, 0.16, 0.47, 0.57, -0.07, -0.68, 0.02, 0.17, 0.21, -0.41, 0.34, -0.66, 0.19, 0.25, 0.1, 0.01, 0.7, -0.13, 0.08, -0.08, 0.1, 0.01, 0.03, 0.18, -0.36 }, {-0.02, 0.16, 0.15, 0.2, 0.25, -0.64, -0.41, 0.26, 0.12, 0.05, 0.19, 0.03, 0.1, 0.5, -0.14, 0.25, 0.39, 0.3, 0.36, -0.26, -0.05, -0.16, -0.2, -0.41, -0.13, 0.0, 0.27, -0.19, -0.43, -0.17, 0.1, -0.21, 0.0, -0.01, -0.33, -0.16, 0.15, 0.14, 0.48, -0.09, -0.37, -0.01, -0.05, 0.01, -0.15, -0.0, -0.68, -0.15, 0.29, 0.02, 0.12, -0.03, 0.11, -0.34, -0.33, 0.34, -0.05, 0.2, 0.26, -0.33 }, {-0.17, -0.03, 0.11, 0.21, -0.01, -0.12, -0.68, 0.2, 0.16, 0.07, 0.14, 0.05, 0.09, 0.47, -0.32, 0.12, 0.15, 0.23, -0.05, 0.16, 0.03, -0.14, -0.15, -0.34, 0.53, -0.23, -0.09, -0.2, 0.28, 0.17, 0.44, -0.12, 0.11, -0.2, -0.37, 0.51, 0.11, 0.02, 0.3, 0.16, 0.16, -0.01, 0.31,

0.18, 0.25, -0.16, -0.37, -0.07, -0.1, 0.09, 0.26, -0.37, 0.08, 0.51, 0.21, -0.37, 0.62, 0.05, 0.11, -0.22 }, {-0.25, -0.15, -0.07, 0.29, -0.05, -0.07, -0.51, -0.15, 0.23, 0.13, 0.22, 0.12, 0.12, 0.17, -0.29, 0.52, -0.06, 0.23, -0.39, 0.05, 0.11, 0.1, -0.33, -0.08, 0.45, -0.23, -0.23, -0.04, 0.39, -0.34, 0.6, -0.06, 0.17, 0.21, -0.29, 0.52, 0.14, 0.09, 0.0, 0.18, -0.11, -0.11, 0.21, 0.34, 0.16, -0.02, -0.28, 0.14, -0.27, 0.0, -0.22, -0.01, 0.1, 0.54, 0.0, -0.68, 0.5, 0.01, 0.14, -0.43 }, {-0.33, -0.04, -0.18, 0.02, -0.14, 0.01, 0.0, -0.68, -0.02, 0.05, 0.19, 0.12, 0.12, 0.24, -0.08, -0.03, -0.22, 0.13, -0.21, -0.43, 0.11, 0.52, -0.26, 0.22, 0.06, -0.41, 0.08, -0.07, 0.19, -0.57, 0.06, -0.01, 0.15, 0.28, -0.29, 0.43, 0.05, 0.19, -0.29, 0.08, -0.47, 0.02, 0.26, 0.09, 0.2, 0.19, -0.32, -0.14, -0.43, 0.04, 0.19, 0.65, 0.31, 0.64, 0.22, -0.43, 0.09, -0.37, 0.05, -0.2 }, {0.11, 0.09, -0.03, 0.27, -0.07, -0.02, 0.03, -0.31, -0.06, 0.04, 0.09, 0.04, 0.06, 0.3, -0.14, -0.01, -0.02, 0.18, -0.13, -0.09, 0.17, 0.37, -0.22, 0.24, -0.15, -0.28, -0.04, 0.04, 0.12, -0.36, -0.24, -0.02, 0.31, 0.4, -0.2, 0.2, 0.04, -0.03, -0.42, 0.01, -0.47, 0.01, 0.09, -0.02, 0.22, 0.26, -0.3, -0.16, -0.37, 0.1, 0.33, 0.53, 0.22, 0.57, 0.29, -0.15, -0.17, -0.43, -0.04, -0.0 }, {-0.29, 0.1, -0.1, 0.12, -0.15, 0.18, 0.08, -0.08, -0.01, 0.04, -0.1, 0.04, 0.02, 0.16, -0.15, 0.11, 0.09, 0.3, -0.14, -0.12, 0.09, 0.07, -0.15, 0.24, 0.07, -0.45, 0.18, -0.0, 0.05, -0.25, -0.31, -0.04, 0.17, 0.18, -0.13, 0.24, 0.02, -0.21, -0.29, -0.21, -0.12, -0.15, -0.22, -0.09, 0.07, 0.11, -0.22, -0.24, -0.25, 0.04, 0.24, 0.25, -0.03, 0.36, 0.03, 0.09, 0.05, -0.51, -0.2, 0.24 }, {-0.2, 0.08, -0.08, -0.31, 0.2, -0.03, -0.19, 0.07, 0.22, 0.08, -0.57, -0.0, 0.02, -0.23, -0.25, -0.32, 0.32, -0.14, -0.01, -0.1, 0.45, 0.21, 0.17, -0.1, 0.13, -0.05, 0.42, 0.0, -0.12, 0.18, 0.05, 0.08, -0.09, -0.59, -0.07, 0.18, 0.04, 0.28, 0.23, 0.16, 0.17, -0.06, -0.4, 0.07, -0.21, 0.39, 0.04, -0.1, 0.13, 0.15, 0.45, 0.1, -0.41, -0.07, 0.31, 0.14, -0.11, -0.09, -0.48, 0.21 }, {-0.55, 0.03, -0.0, -0.09, 0.41, -0.05, 0.03, -0.08, 0.04, 0.08, -0.43, 0.02, -0.02, -0.14, -0.07, -0.16, 0.31, -0.05, -0.07, 0.23, 0.59, 0.38, -0.11, 0.05, 0.8, 0.03, 0.76, 0.05, 0.29, 0.06, -0.26, -0.02, 0.09, -0.15, 0.03, 0.59, -0.1, 0.06, 0.2, 0.06, -0.39, -0.21, -0.36, 0.1, -0.05, 0.22, 0.15, -0.15, -0.11, 0.01, 0.05,

-0.01, -0.46, -0.23, 0.23, -0.13, 0.05, -0.25, -0.41, -0.1 }, {-0.27, -0.22, 0.05, -0.31, 0.18, 0.2, -0.12, -0.05, 0.16, 0.08, -0.34, 0.04, 0.0, -0.1, -0.05, 0.1, 0.14, 0.16, -0.07, 0.12, 0.47, 0.1, -0.03, -0.05, 0.56, -0.08, 0.61, -0.16, 0.07, 0.35, -0.37, 0.01, 0.07, -0.04, -0.01, 0.08, 0.02, 0.35, -0.12, -0.03, -0.09, -0.14, -0.02, -0.05, -0.06, 0.39, 0.21, -0.13, -0.31, 0.03, -0.1, -0.02, -0.49, -0.01, 0.32, 0.13, -0.03, -0.44, -0.23, 0.14 }, {-0.14, 0.04, 0.09, 0.03, 0.29, -0.13, -0.16, 0.14, -0.02, 0.05, -0.24, 0.02, -0.02, 0.24, 0.07, 0.11, 0.41, 0.02, -0.05, 0.04, 0.02, -0.2, -0.22, -0.27, 0.14, -0.09, 0.13, -0.11, -0.03, -0.06, -0.22, 0.03, -0.16, 0.13, -0.11, 0.11, -0.13, 0.15, -0.37, -0.27, -0.25, -0.14, 0.03, -0.39, 0.05, 0.16, -0.13, -0.13, -0.35, -0.03, 0.01, 0.03, -0.26, 0.21, 0.02, 0.09, 0.49, -0.56, -0.2, -0.27 }, {-0.18, -0.2, 0.12, 0.4, 0.38, 0.52, 0.18, -0.12, 0.03, -0.06, -0.35, 0.02, -0.05, 0.25, 0.12, 0.01, 0.12, -0.35, -0.05, -0.06, -0.53, 0.09, -0.1, -0.0, -0.12, -0.04, -0.06, 0.05, -0.09, -0.62, -0.52, -0.03, 0.18, 0.18, 0.21, 0.02, -0.14, -0.02, -0.31, -0.36, -0.21, -0.27, 0.05, -0.39, 0.12, 0.05, -0.12, -0.17, -0.31, -0.04, -0.03, 0.03, 0.29, 0.07, -0.12, 0.5, 0.14, -0.51, -0.06, -0.25 }, {0.47, 0.3, 0.02, -0.2, -0.17, 0.67, -0.07, 0.68, 0.05, -0.03, -0.2, 0.07, 0.04, -0.31, -0.52, -0.85, 0.13, 0.15, -0.01, 0.45, -0.33, 0.18, -0.23, 0.0, -0.58, -0.58, -0.11, 0.36, -0.02, 0.66, -0.93, -0.41, 0.07, -0.4, 0.01, -0.51, -0.06, -0.97, -0.41, 0.58, 0.82, -0.08, 0.02, -0.8, -0.2, -0.11, 0.26, 0.26, 0.48, -0.01, 0.26, -0.77, -0.58, 0.23, 0.08, -0.82, -0.92, 0.01, -0.3, 0.38 }, {0.5, 0.31, -0.13, -0.43, -0.24, 0.59, -0.06, 0.69, -0.02, 0.03, -0.12, 0.11, 0.02, -0.23, -0.58, -0.81, 0.11, 0.28, -0.05, 0.37, -0.43, 0.15, -0.34, 0.02, -0.45, -0.53, -0.05, 0.23, 0.05, 0.62, -0.66, -0.39, 0.06, -0.35, 0.02, -0.55, -0.05, -0.82, -0.3, 0.59, 0.71, 0.04, -0.02, -0.58, -0.28, -0.16, 0.15, 0.27, 0.55, -0.06, 0.11, -0.71, -0.51, 0.18, 0.05, -0.82, -0.95, 0.11, -0.33, 0.36 }, {0.41, 0.53, 0.11, -0.01, -0.05, 0.18, 0.14, 0.75, 0.06, 0.05, -0.07, -0.0, 0.05, 0.37, -0.23, -0.43, 0.32, 0.59, 0.37, 0.34, 0.24, 0.08, -0.31, -0.04, -0.58, -0.52, 0.48, 0.3, -0.34, 0.73, -0.25, -0.22, 0.14, -0.24, -0.23, -0.43, 0.0, -0.74, -0.2, 0.19, -0.01, -0.04, -0.39, -0.37, -0.22, -0.3, -0.38, 0.01, 0.3, -0.01, 0.27, -0.23, -0.55, -0.14, 0.02, -0.27,

-0.55, -0.27, -0.46, 0.48 }, {0.27, 0.49, -0.14, 0.23, -0.06, -0.22, 0.15, 0.19, 0.26, 0.13, 0.29, 0.02, 0.07, 0.23, 0.25, -0.27, 0.57, 0.34, 0.45, -0.18, 0.8, 0.12, -0.76, 0.07, -0.39, -0.34, 0.54, -0.14, -0.39, 0.48, -0.3, -0.28, 0.35, -0.26, -0.05, -0.28, -0.02, -0.08, -0.09, 0.2, -0.74, -0.15, -0.46, -0.34, -0.33, 0.07, -0.52, 0.11, 0.34, 0.17, -0.01, 0.12, -1.02, -0.1, -0.08, -0.09, -0.27, -0.22, -0.62, 0.4 }, {0.01, 0.44, 0.16, 0.01, 0.04, -0.24, -0.63, 0.45, 0.29, 0.11, 0.26, 0.02, 0.14, -0.18, -0.35, -0.29, 0.6, -0.15, 0.19, -0.48, 0.66, 0.37, -0.17, -0.09, 0.42, -0.11, 0.38, -0.15, -0.29, -0.03, -0.26, -0.68, 0.32, -0.1, -0.2, -0.13, 0.16, 0.37, -0.01, 0.68, -0.54, -0.04, 0.25, -0.24, -0.74, 0.55, -0.14, 0.48, 0.32, 0.09, -0.44, 0.52, -0.19, 0.14, 0.17, -0.59, -0.28, 0.2, -0.27, 0.05 }, {-0.19, 0.21, 0.08, 0.16, -0.21, -0.13, -0.23, 0.02, 0.22, 0.03, 0.38, 0.04, 0.04, -0.19, -0.19, 0.01, 0.24, 0.11, 0.15, -0.77, 0.39, 0.15, -0.2, 0.0, 0.12, 0.07, 0.18, -0.15, -0.24, -0.29, 0.08, 0.04, 0.17, 0.07, -0.22, 0.2, 0.09, 0.34, -0.15, 0.41, -0.68, -0.13, 0.25, 0.02, -0.36, 0.19, -0.25, 0.06, 0.29, 0.11, -0.34, 0.42, 0.18, -0.01, -0.01, -0.3, 0.14, 0.12, -0.03, 0.01 }, {0.05, 0.17, 0.24, 0.2, -0.29, 0.15, -0.54, 0.27, 0.09, 0.03, 0.47, 0.01, 0.03, 0.07, -0.15, -0.29, 0.13, 0.09, 0.02, -0.49, 0.32, 0.09, 0.06, -0.05, 0.43, -0.51, -0.2, -0.18, 0.29, -0.16, -0.14, -0.16, 0.03, -0.2, -0.65, 0.33, 0.16, -0.06, -0.23, 0.36, -0.36, -0.13, 0.23, 0.08, 0.4, -0.04, -0.26, -0.08, 0.22, 0.01, 0.01, -0.14, 0.24, 0.2, 0.31, -0.78, 0.06, 0.06, -0.09, 0.2 }, {-0.09, 0.27, 0.12, 0.36, -0.02, -0.16, -0.46, 0.43, 0.13, 0.08, 0.26, 0.05, 0.1, 0.46, -0.38, -0.18, -0.03, -0.26, 0.04, 0.04, 0.11, 0.31, -0.16, -0.08, 0.47, -0.26, -0.33, -0.17, 0.36, -0.01, -0.14, -0.06, 0.03, -0.15, -0.34, 0.36, 0.16, -0.1, 0.12, 0.34, -0.14, -0.29, 0.01, 0.04, 0.16, 0.04, -0.4, 0.13, 0.31, 0.04, -0.12, -0.03, 0.23, 0.49, 0.08, -0.48, 0.16, 0.13, -0.02, -0.13 }, {-0.42, 0.3, -0.16, -0.01, 0.21, 0.02, -0.38, -0.05, 0.08, 0.09, 0.16, 0.12, 0.05, 0.73, -0.39, -0.77, -0.17, -0.26, 0.02, -0.32, 0.27, 0.8, -0.24, 0.16, 0.3, -0.72, -0.24, 0.03, 0.28, -0.3, 0.39, -0.45, 0.42, -0.18, -0.25, 0.45, -0.05, 0.03, 0.17, 0.55, -0.02, 0.06, 0.04, 0.2, 0.03, 0.14, -0.4, -0.03, -0.19, 0.1, -0.08, 0.39, 0.38, 0.64, 0.37, -0.69, -0.11, -0.19, -0.08, 0.03 }, {-0.14, 0.4, -0.11,

0.06, 0.34, -0.15, -0.28, 0.08, -0.03, 0.01, 0.09, 0.07, 0.04, 0.49, -0.07, -0.6, -0.17, -0.29, 0.05, -0.26, 0.27, 0.53, -0.14, 0.25, 0.25, -0.31, -0.15, 0.1, 0.15, -0.28, 0.21, -0.28, 0.27, -0.06, -0.19, 0.27, 0.03, 0.17, 0.14, 0.22, -0.13, -0.01, -0.12, 0.12, -0.01, 0.06, -0.36, -0.12, -0.28, 0.03, 0.06, 0.47, 0.4, 0.3, 0.26, -0.19, 0.02, -0.15, -0.12, 0.13 }, {-0.42, 0.09, -0.21, -0.39, -0.3, -0.07, -0.12, 0.02, -0.05, -0.04, 0.05, 0.04, 0.04, 0.39, -0.2, -0.55, -0.21, 0.04, -0.11, -0.04, 0.33, 0.61, -0.39, 0.15, 0.29, -0.74, -0.03, -0.02, 0.28, 0.09, 0.11, -0.06, -0.19, -0.7, -0.16, 0.51, -0.08, -0.33, -0.34, 0.38, -0.02, 0.06, 0.09, 0.03, -0.1, -0.01, 0.03, -0.19, -0.49, 0.04, 0.42, 0.06, 0.21, 0.76, 0.33, -0.76, -0.1, -0.32, -0.29, 0.74 }, {-0.08, 0.3, -0.14, -0.62, 0.59, -0.43, -0.4, 0.44, 0.19, -0.03, -0.48, 0.07, 0.01, 0.06, -0.25, -0.79, 0.32, -0.56, 0.25, 0.31, 0.63, 0.54, -0.02, -0.15, 0.3, -0.11, 0.09, 0.08, 0.19, 0.5, -0.37, -0.17, -0.32, -1.28, 0.01, 0.15, -0.11, 0.16, 0.58, 0.61, 0.54, 0.09, -0.45, -0.0, -0.35, 0.29, 0.33, -0.05, 0.3, 0.0, 0.69, -0.18, -0.27, -0.41, 0.65, -0.08, -0.68, 0.42, -0.62, 0.65 }, {-0.15, -0.03, -0.15, -0.2, 0.22, 0.1, 0.01, 0.28, 0.1, 0.02, -0.49, 0.01, 0.08, -0.22, -0.12, -0.35, 0.45, -0.38, -0.26, 0.31, 0.52, 0.47, -0.17, 0.27, 0.57, 0.03, 0.56, 0.27, 0.22, 0.18, -0.65, -0.17, 0.14, -0.07, -0.26, 0.17, -0.11, -0.07, 0.12, 0.02, -0.28, -0.02, -0.39, -0.26, 0.02, 0.28, 0.28, 0.01, 0.1, -0.01, 0.13, 0.32, -0.24, -0.08, 0.54, -0.13, -0.44, -0.3, -0.53, -0.11 }, {-0.13, -0.09, -0.17, -0.16, 0.14, 0.55, 0.04, 0.06, 0.08, -0.0, -0.56, 0.09, 0.04, -0.4, 0.05, -0.16, 0.16, -0.03, -0.26, 0.35, 0.48, 0.28, 0.06, 0.25, 0.31, -0.29, 0.47, 0.11, 0.02, 0.25, -0.38, 0.17, 0.27, -0.14, -0.15, 0.02, -0.05, -0.06, -0.32, -0.19, 0.04, -0.08, -0.36, -0.08, 0.11, 0.29, 0.52, -0.32, -0.24, 0.05, 0.27, -0.05, -0.25, 0.15, 0.44, -0.17, -0.18, -0.5, -0.35, 0.14 }, {-0.43, 0.01, 0.01, 0.01, 0.28, 0.19, -0.24, 0.13, 0.04, -0.03, -0.33, -0.01, -0.05, -0.43, 0.27, -0.33, -0.02, 0.25, -0.14, 0.46, 0.26, 0.08, 0.1, -0.19, 0.22, -0.21, 0.34, 0.31, 0.06, 0.09, 0.11, 0.39, 0.1, -0.35, -0.05, 0.3, -0.09, -0.22, -0.55, -0.31, -0.03, -0.14, -0.32, -0.09, 0.29, 0.15, 0.2, -0.27, -0.39, -0.08, 0.56, -0.2, -0.39, 0.39, 0.2, -0.25, 0.29, -0.38, -0.5, 0.22 }, {-0.2, -0.25, 0.06, 0.07, 0.33, 0.5, -0.18, 0.15, -

0.02, -0.01, -0.47, 0.04, -0.03, -0.21, 0.08, -0.41, 0.11, -0.01, -0.08, 0.25, -0.18, 0.37, 0.17, 0.12, -0.02, -0.12, 0.23, 0.48, -0.12, -0.27, -0.18, 0.08, 0.04, -0.02, 0.1, 0.13, 0.03, -0.34, -0.49, -0.09, 0.07, -0.07, 0.03, -0.16, 0.27, 0.15, 0.29, -0.21, -0.21, -0.04, 0.41, -0.04, 0.01, 0.29, 0.24, -0.2, -0.48, -0.48, -0.31, 0.02 }, {0.3, 0.3, -0.27, -0.07, -0.35, 0.43, -0.02, 0.75, 0.0, 0.01, -0.15, 0.09, -0.02, 0.14, -0.36, -0.9, 0.03, -0.14, 0.16, 0.61, -0.52, 0.0, -0.36, -0.1, -0.39, -0.75, -0.02, -0.01, 0.21, 0.9, -0.93, -0.53, -0.39, -0.23, 0.42, -0.68, -0.02, -1.17, -0.37, 0.68, 0.76, 0.12, -0.26, -0.82, -0.32, -0.21, 0.22, 0.28, 0.42, 0.01, 0.06, -0.71, -0.74, 0.31, -0.0, -0.52, -0.71, 0.12, -0.39, 0.05 }, {0.42, 0.33, -0.24, -0.17, -0.28, 0.4, 0.02, 0.78, -0.03, -0.02, -0.11, 0.12, -0.09, 0.22, -0.38, -0.87, 0.06, -0.07, 0.05, 0.61, -0.61, -0.01, -0.42, -0.06, -0.43, -0.77, -0.08, 0.09, 0.11, 0.93, -0.82, -0.55, -0.38, -0.31, 0.38, -0.66, -0.02, -1.09, -0.36, 0.69, 0.74, 0.19, -0.29, -0.75, -0.36, -0.19, 0.2, 0.2, 0.41, -0.03, 0.07, -0.63, -0.79, 0.2, -0.03, -0.54, -0.67, 0.14, -0.35, 0.0 }, {0.73, 0.41, 0.16, -0.1, -0.44, 0.17, 0.04, 0.63, 0.08, 0.01, -0.01, 0.13, -0.05, 0.51, -0.04, -0.42, 0.28, 0.14, 0.44, 0.21, -0.06, -0.05, -0.35, -0.23, -0.42, -0.49, 0.53, 0.23, -0.21, 0.73, -0.6, -0.42, 0.26, 0.11, 0.1, -0.74, -0.01, -0.44, 0.05, 0.11, 0.32, 0.12, -0.22, -0.76, -0.39, -0.27, -0.28, 0.1, 0.71, 0.01, 0.04, -0.39, -0.7, -0.39, -0.08, 0.27, -0.67, 0.16, -0.5, 0.13 }, {0.47, 0.05, -0.08, -0.09, -0.14, -0.19, 0.16, 0.25, 0.29, 0.02, 0.14, 0.15, 0.17, -0.03, 0.29, 0.01, 0.48, 0.13, 0.19, 0.2, 0.44, -0.11, -0.51, -0.06, -0.15, -0.33, 0.78, -0.22, -0.35, 0.59, -0.53, -0.36, 0.31, -0.11, 0.24, -0.38, -0.0, -0.12, 0.12, 0.04, -0.26, 0.06, -0.34, -0.28, -0.45, -0.05, -0.22, -0.0, 0.22, 0.09, 0.14, -0.35, -1.09, -0.34, -0.22, 0.12, -0.44, -0.16, -0.62, 0.23 }, {0.21, 0.59, 0.26, -0.07, 0.49, -0.38, -0.47, 0.83, 0.39, 0.07, 0.11, 0.08, 0.08, -0.05, -0.21, -0.16, 0.47, -0.49, 0.01, 0.23, 0.31, 0.17, 0.09, -0.11, 0.01, 0.01, 0.02, 0.06, -0.12, 0.22, -0.44, -0.77, 0.07, -0.51, 0.03, 0.03, 0.2, 0.34, 0.2, 0.32, 0.22, -0.03, -0.19, -0.27, -0.55, 0.12, -0.26, 0.05, 0.43, 0.1, -0.07, -0.7, -0.0, -0.49, 0.02, -0.26, -0.43, 0.44, -0.45, 0.22 }, {0.03, 0.3, -0.05, 0.02, 0.27, -0.39, -0.09, 0.28, 0.16, 0.01, 0.5, 0.02, 0.09, -0.12, 0.06,

0.33, 0.13, -0.12, -0.11, -0.4, 0.44, 0.02, -0.16, -0.07, 0.03, 0.07, -0.06, -0.25, -0.03, -0.15, 0.0, -0.16, 0.04, -0.15, -0.11, 0.24, 0.15, 0.43, -0.12, -0.07, -0.21, 0.09, -0.03, 0.07, -0.35, -0.06, -0.4, -0.15, -0.1, 0.15, -0.11, -0.25, 0.09, -0.24, -0.12, -0.29, 0.47, 0.01, -0.23, 0.21 }, {0.1, -0.11, 0.14, 0.57, -0.1, -0.08, -0.25, -0.08, 0.1, 0.03, 0.43, 0.13, 0.07, -0.2, 0.05, 0.38, -0.18, 0.15, -0.09, -0.73, 0.03, -0.15, 0.57, -0.2, -0.11, -0.2, -0.27, 0.08, 0.14, -0.24, 0.05, 0.17, 0.06, 0.39, -0.31, 0.49, 0.16, 0.21, 0.02, 0.13, -0.1, 0.09, 0.42, 0.21, 0.29, 0.15, -0.04, -0.15, 0.16, 0.13, 0.03, 0.05, 0.32, -0.25, -0.16, -0.34, 0.06, 0.06, 0.22, -0.02 }, {0.22, 0.31, 0.09, 0.4, 0.14, -0.42, -0.3, 0.39, 0.14, 0.13, 0.38, 0.08, 0.13, 0.32, -0.18, 0.1, -0.4, -0.14, 0.25, -0.27, -0.29, -0.1, -0.06, -0.19, 0.08, 0.1, -0.41, 0.03, 0.36, -0.07, -0.22, 0.28, 0.09, 0.25, -0.03, 0.13, 0.13, 0.19, 0.27, 0.4, -0.18, -0.24, -0.14, -0.07, 0.06, 0.07, -0.43, 0.17, 0.39, 0.07, -0.35, 0.15, 0.2, 0.02, -0.39, 0.15, 0.03, 0.45, 0.18, -0.39 }, {-0.06, 0.45, -0.13, 0.09, 0.23, -0.1, -0.26, -0.11, 0.11, 0.13, 0.42, 0.07, 0.07, 0.58, -0.62, -0.15, -0.32, -0.43, 0.26, -0.87, 0.1, 0.36, -0.19, 0.05, 0.21, -0.3, -0.09, 0.08, 0.08, -0.59, -0.12, -0.19, 0.63, 0.33, -0.17, 0.21, 0.01, 0.53, 0.82, 0.63, -0.31, 0.22, -0.06, 0.07, -0.19, 0.26, -0.44, -0.04, 0.09, 0.11, -0.1, 0.8, 0.21, 0.23, -0.2, 0.06, -0.45, -0.09, 0.03, 0.02 }, {-0.09, 0.49, -0.26, 0.2, 0.39, -0.01, -0.34, 0.22, 0.01, 0.03, 0.37, 0.02, 0.13, 0.42, -0.32, -0.18, -0.42, -0.43, 0.44, -0.47, -0.07, 0.22, -0.5, 0.03, 0.21, -0.17, 0.11, 0.03, 0.3, -0.29, -0.05, -0.17, 0.47, 0.25, 0.01, 0.2, -0.08, 0.29, 0.54, 0.51, -0.19, 0.03, -0.22, -0.07, -0.15, 0.05, -0.36, 0.27, 0.18, 0.14, -0.09, 0.31, 0.1, 0.07, -0.19, -0.01, -0.2, 0.03, -0.03, 0.01 }, {-0.23, 0.32, -0.15, -0.15, -0.08, 0.14, -0.12, -0.07, -0.1, -0.02, 0.08, 0.02, 0.05, 0.14, -0.05, -0.2, -0.33, 0.13, 0.08, -0.4, -0.11, 0.41, -0.29, 0.23, 0.16, -0.42, 0.17, 0.16, 0.35, -0.09, 0.02, 0.03, -0.18, -0.3, -0.0, 0.58, -0.19, -0.03, 0.15, 0.15, -0.09, -0.03, -0.26, 0.15, -0.0, -0.04, -0.08, -0.08, 0.08, 0.01, 0.43, 0.22, 0.32, 0.01, -0.11, -0.28, -0.09, -0.01, -0.05, 0.53 }, {0.33, 0.1, -0.27, -0.27, 0.1, -0.3, -0.33, 0.46, 0.07, 0.07, 0.31, 0.03, 0.06, 0.38, -0.05, -0.02, 0.24, 0.1, 0.21, -0.06, 0.43, 0.27, -0.11, -0.22, 0.27,

-0.28, 0.1, -0.26, 0.08, 0.33, -0.26, -0.06, -0.3, -0.76, 0.08, -0.34, -0.16, 0.31, 0.01, 0.36, 0.47, 0.04, -0.25, 0.03, -0.32, 0.16, 0.04, 0.03, 0.13, -0.0, 0.19, 0.04, -0.25, -0.46, 0.15, -0.06, -0.17, 0.36, -0.46, 0.52 }, {0.35, -0.07, -0.31, 0.05, -0.24, -0.01, 0.33, 0.28, 0.12, 0.02, -0.05, 0.13, 0.11, -0.11, -0.05, -0.04, 0.45, 0.06, -0.17, 0.12, 0.41, 0.45, -0.32, 0.29, 0.45, -0.09, 0.43, -0.14, 0.18, 0.07, -0.64, 0.02, 0.15, 0.22, -0.06, -0.23, -0.1, -0.37, -0.51, -0.09, -0.07, -0.06, -0.16, -0.36, -0.02, 0.21, 0.21, 0.04, -0.08, 0.09, -0.23, 0.32, -0.15, 0.13, 0.37, -0.14, -0.43, -0.32, -0.3, -0.27 }, {0.11, -0.3, -0.2, 0.13, -0.11, 0.38, 0.27, 0.11, 0.08, 0.06, -0.08, 0.01, 0.05, -0.27, 0.14, -0.01, 0.23, 0.56, -0.41, 0.31, 0.52, 0.22, 0.13, 0.2, -0.0, -0.28, 0.55, -0.06, -0.07, -0.2, -0.29, 0.08, 0.29, 0.07, -0.07, -0.13, 0.04, -0.23, -0.68, -0.39, 0.2, -0.05, -0.1, -0.1, 0.03, 0.33, 0.41, -0.32, -0.54, 0.12, 0.11, -0.06, -0.09, 0.25, 0.19, -0.04, -0.24, -0.47, -0.25, -0.01 }, {-0.51, -0.35, -0.05, 0.32, 0.04, 0.19, 0.0, -0.27, 0.08, -0.02, -0.42, 0.06, -0.05, -0.31, 0.24, -0.1, -0.05, 0.27, -0.47, 0.36, 0.37, 0.17, 0.24, -0.15, 0.44, -0.12, 0.39, 0.34, 0.24, -0.34, 0.43, 0.26, 0.17, -0.2, -0.08, 0.52, -0.06, -0.18, -0.51, -0.27, -0.04, 0.01, -0.13, 0.15, 0.21, 0.2, 0.43, -0.32, -0.47, 0.04, 0.49, -0.1, -0.34, 0.28, 0.17, -0.31, 0.5, -0.51, -0.34, 0.09 }, {-0.27, -0.5, -0.09, 0.0, 0.0, 0.33, -0.03, -0.14, 0.08, -0.04, -0.36, 0.08, -0.04, -0.58, 0.04, -0.49, -0.09, 0.17, -0.58, 0.28, 0.28, 0.5, 0.23, -0.04, 0.14, -0.22, 0.24, 0.4, -0.0, -0.38, 0.15, 0.21, 0.18, -0.36, 0.02, 0.52, -0.06, -0.5, -0.41, -0.25, 0.15, -0.01, -0.1, 0.07, 0.25, 0.27, 0.55, -0.46, -0.54, 0.0, 0.64, -0.21, -0.06, 0.31, 0.25, -0.41, -0.24, -0.33, -0.46, 0.33 }, {0.52, -0.02, -0.26, 0.12, -0.42, 0.37, -0.13, 0.55, -0.03, 0.01, 0.04, 0.08, 0.03, 0.34, -0.38, -0.59, 0.08, -0.1, 0.41, 0.46, -0.46, -0.08, -0.35, -0.03, -0.23, -0.47, -0.15, -0.12, -0.07, 0.65, -0.42, -0.24, -0.33, -0.22, 0.4, -0.61, 0.02, -0.56, -0.29, 0.46, 0.67, 0.38, -0.07, -0.66, 0.02, -0.17, 0.15, 0.16, 0.58, -0.01, -0.28, -0.39, -0.18, 0.34, -0.01, -0.28, -0.73, 0.08, -0.13, 0.03 }, {0.45, 0.06, -0.25, 0.2, -0.4, 0.45, -0.15, 0.5, 0.08, -0.01, 0.1, 0.12, 0.02, 0.37, -0.33, -0.67, 0.08, -0.2, 0.39, 0.44, -0.45, -0.1, -0.29, -0.05, -0.29, -0.51, -0.12, -0.11, -0.06, 0.59, -0.37, -0.33, -

0.25, -0.26, 0.3, -0.58, 0.09, -0.66, -0.28, 0.44, 0.61, 0.47, -0.15, -0.65, 0.01, -0.12, 0.17, 0.21, 0.64, 0.06, -0.37, -0.36, -0.22, 0.31, -0.03, -0.37, -0.65, 0.1, -0.07, -0.0 }, {0.88, 0.46, 0.01, -0.05, -0.46, 0.28, -0.1, 0.79, 0.05, 0.04, 0.27, 0.02, 0.03, 0.56, -0.3, -0.21, 0.21, -0.2, 0.57, 0.44, -0.37, -0.09, -0.73, -0.1, -0.41, -0.4, 0.37, -0.09, -0.3, 0.81, -0.9, -0.2, 0.53, -0.06, 0.45, -1.03, -0.0, -0.37, -0.23, 0.33, 0.43, 0.47, -0.32, -1.18, -0.19, -0.2, -0.18, 0.44, 1.07, 0.05, -0.27, -0.65, -0.34, -0.1, -0.07, 0.33, -1.05, 0.49, -0.21, 0.42 }, {0.67, 0.28, -0.39, 0.38, -0.44, 0.23, 0.42, 0.44, 0.13, 0.11, 0.56, 0.05, 0.05, 0.21, 0.08, 0.19, 0.24, -0.07, 0.41, 0.63, -0.17, -0.29, -0.72, 0.17, -0.55, -0.5, 0.41, -0.53, -0.51, 0.65, -0.72, 0.5, 0.44, -0.01, 0.53, -0.47, 0.12, 0.0, -0.44, -0.25, -0.31, 0.35, -0.38, -0.65, 0.02, -0.18, -0.07, 0.09, 0.31, 0.07, -0.18, -0.57, -0.59, 0.19, -0.41, 0.25, -0.68, -0.08, -0.2, 0.27 }, {0.15, 0.05, -0.56, 0.27, -0.11, -0.04, 0.34, 0.43, 0.32, 0.08, 0.29, 0.1, 0.06, 0.36, -0.34, 0.19, 0.09, -0.27, -0.06, 0.18, -0.4, -0.06, 0.25, 0.25, 0.21, -0.52, -0.06, -0.11, -0.16, 0.2, -0.42, -0.03, 0.2, -0.01, 0.3, -0.21, 0.02, 0.24, -0.27, -0.4, 0.26, 0.35, -0.17, -0.25, -0.22, 0.09, 0.11, 0.15, 0.15, 0.06, -0.42, -0.12, 0.28, 0.09, -0.03, -0.28, -0.44, 0.38, -0.2, 0.13 }, {0.44, 0.49, -0.86, -0.04, -0.04, 0.09, 0.43, 0.34, 0.33, 0.03, 0.45, 0.1, 0.06, 0.55, -0.25, 0.31, 0.03, 0.02, 0.41, 0.15, -0.29, -0.42, -0.49, 0.14, 0.24, -0.43, -0.22, -0.49, 0.04, 0.61, -0.6, 0.46, 0.05, -0.05, 0.39, -0.53, 0.1, -0.45, -0.61, -0.5, 0.22, 0.27, -0.39, -0.36, -0.0, 0.08, -0.25, 0.18, 0.38, 0.11, -0.42, 0.12, 0.25, 0.27, -0.36, 0.12, -0.36, 0.37, -0.31, 0.57 }, {0.12, 0.27, -0.29, 0.24, -0.16, 0.16, 0.07, -0.66, 0.25, 0.05, 0.55, 0.12, 0.17, 0.2, -0.2, 0.28, -0.14, -0.22, 0.03, -0.57, -0.13, -0.23, -0.17, -0.01, 0.21, -0.3, -0.23, -0.07, 0.25, -0.03, 0.41, 0.31, 0.3, 0.93, -0.04, 0.21, 0.06, 0.24, 0.32, 0.07, -0.25, 0.17, 0.04, 0.43, 0.12, 0.21, -0.02, 0.38, 0.06, 0.07, -0.13, 0.95, 0.07, 0.17, 0.01, -0.07, 0.52, 0.11, 0.11, -0.24 }, {-0.0, 0.2, -0.25, 0.68, -0.13, 0.0, 0.2, -0.4, 0.2, 0.12, 0.62, 0.08, 0.04, 0.39, -0.56, 0.29, 0.03, 0.22, -0.12, -0.36, 0.1, -0.25, -0.23, -0.08, 0.21, -0.33, -0.34, -0.21, 0.38, -0.29, 0.45, -0.2, 0.42, 1.0, -0.08, 0.35, 0.01, 0.17, 0.29, 0.05, -0.6, -0.0, -0.21, 0.48,

-0.03, 0.17, -0.23, 0.34, -0.11, 0.01, -0.14, 0.79, -0.13, 0.2, 0.0, -0.21, 0.28, -0.23, 0.37, -0.75 }, {0.01, 0.58, -0.0, 0.24, -0.04, -0.48, 0.02, -0.3, -0.04, 0.05, 0.62, 0.11, 0.02, 0.43, -0.43, -0.07, -0.13, 0.23, 0.14, -0.54, 0.28, -0.04, -0.08, -0.15, 0.27, -0.52, -0.38, -0.37, 0.59, -0.52, 0.71, -0.0, 0.28, 0.78, -0.23, 0.48, 0.15, 0.19, 0.39, 0.62, -0.34, 0.24, 0.27, 0.5, 0.22, 0.33, -0.36, 0.11, 0.15, 0.14, -0.38, 0.82, 0.01, 0.72, -0.12, -0.57, -0.27, -0.3, 0.21, -0.7 }, {-0.05, 0.64, -0.16, -0.01, 0.08, -0.51, 0.08, -0.11, -0.11, -0.01, 0.77, 0.11, 0.05, 0.47, -0.23, 0.18, -0.29, 0.01, 0.55, -0.19, 0.21, -0.36, -0.66, -0.18, 0.25, -0.11, -0.33, -0.41, 0.82, -0.05, -0.06, 0.22, 0.33, 0.67, 0.02, 0.09, 0.02, 0.12, 0.13, 0.55, -0.15, 0.01, 0.06, 0.01, 0.07, 0.04, -0.51, 0.48, 0.21, 0.07, -0.19, 0.41, -0.26, 0.31, -0.2, -0.13, 0.41, -0.11, -0.07, -0.59 }, {-0.45, 0.23, 0.24, 0.42, -0.05, 0.52, 0.14, -0.34, 0.02, 0.08, 0.1, 0.0, 0.09, -0.05, -0.06, -0.14, -0.25, -0.18, -0.08, -0.3, -0.16, -0.09, -0.0, 0.28, 0.2, -0.05, 0.02, -0.12, 0.39, 0.16, 0.22, 0.17, 0.21, 0.61, 0.1, 0.67, 0.08, -0.19, -0.05, -0.06, -0.1, -0.1, 0.06, 0.02, 0.25, 0.03, 0.24, -0.07, -0.13, 0.05, -0.17, 0.4, 0.76, 0.11, -0.39, -0.07, -0.05, -0.45, 0.31, -0.51 }, {-0.19, 0.06, 0.05, 0.03, -0.04, 0.52, 0.19, 0.02, 0.2, 0.1, 0.25, 0.03, 0.11, -0.42, 0.31, 0.11, -0.08, -0.26, -0.34, -0.28, 0.17, -0.17, -0.04, -0.07, -0.08, -0.12, -0.24, -0.2, 0.06, 0.07, 0.02, 0.05, 0.01, -0.21, 0.09, 0.19, 0.03, 0.18, -0.21, -0.01, 0.16, -0.09, -0.07, -0.06, 0.01, 0.1, 0.24, -0.03, -0.41, 0.04, 0.2, -0.12, 0.4, -0.1, -0.12, 0.05, 0.18, -0.16, -0.08, -0.29 }, {-0.22, -0.27, -0.2, 0.05, 0.41, 0.09, 0.36, 0.25, -0.04, 0.03, -0.09, 0.07, -0.03, -0.35, 0.43, -0.13, -0.05, -0.38, -0.28, 0.1, 0.28, 0.03, 0.01, 0.2, -0.09, 0.09, 0.32, 0.19, -0.23, -0.02, -0.36, 0.16, -0.07, -0.46, -0.1, 0.19, -0.01, -0.1, -0.06, -0.26, -0.3, 0.04, -0.57, -0.46, 0.06, -0.25, 0.14, -0.15, 0.02, -0.05, 0.38, -0.22, -0.08, -0.08, 0.16, -0.04, 0.05, 0.04, -0.51, 0.16 }, {-0.39, 0.05, -0.1, -0.03, 0.6, 0.06, 0.1, 0.35, -0.1, 0.04, -0.01, 0.05, -0.03, -0.3, 0.48, -0.21, -0.09, -0.05, -0.09, 0.37, 0.32, -0.18, 0.11, 0.15, -0.15, 0.18, 0.29, 0.21, 0.02, 0.18, -0.23, 0.13, -0.08, -0.36, -0.14, 0.24, -0.0, 0.19, -0.05, -0.27, 0.0, 0.15, -0.55, -0.41, 0.04, -0.13, -0.03, -0.12, 0.1, -0.02, 0.39, -0.26, -0.1, 0.11, 0.37,

0.27, 0.17, 0.16, -0.51, 0.07 }, {-0.48, -0.11, -0.08, 0.27, 0.3, 0.31, -0.05, 0.29, 0.07, 0.05, -0.26, 0.0, 0.04, -0.36, 0.16, -0.05, 0.31, -0.0, -0.51, 0.58, 0.74, -0.03, 0.17, -0.06, 0.18, -0.04, 0.15, 0.33, 0.27, 0.0, -0.26, -0.24, -0.07, -0.48, -0.25, 0.75, 0.05, -0.16, -0.32, 0.24, -0.29, 0.1, -0.35, -0.17, 0.17, 0.14, 0.32, -0.27, -0.3, -0.03, 0.56, -0.31, -0.28, 0.33, 0.42, -0.26, -0.07, -0.38, -0.51, 0.05 }, {-0.53, -0.25, -0.07, -0.23, 0.38, 0.21, -0.03, 0.06, 0.09, 0.05, -0.32, 0.09, 0.11, -0.59, 0.41, -0.35, 0.02, -0.03, -0.74, 0.5, 0.81, 0.5, 0.45, 0.02, 0.12, -0.08, 0.21, 0.53, 0.1, -0.2, -0.03, -0.05, 0.09, -0.6, -0.38, 0.88, 0.09, 0.02, 0.06, 0.3, -0.38, 0.16, -0.32, 0.2, 0.18, 0.18, 0.34, -0.4, -0.6, 0.04, 0.95, -0.3, -0.12, 0.39, 0.59, -0.41, -0.18, -0.07, -0.57, 0.37 }, {0.33, 0.15, -0.18, 0.01, -0.25, 0.16, -0.28, 0.36, 0.04, 0.07, 0.02, 0.09, 0.04, 0.12, -0.1, -0.55, -0.05, -0.21, 0.23, 0.23, -0.28, 0.37, -0.11, 0.06, -0.17, -0.31, -0.03, -0.08, 0.05, 0.05, -0.69, -0.34, -0.32, -0.28, 0.06, -0.04, -0.06, -0.33, -0.18, 0.22, 0.22, 0.29, -0.14, -0.43, -0.07, -0.13, -0.06, 0.03, 0.27, -0.05, 0.11, -0.1, -0.37, 0.28, -0.05, -0.4, -0.53, 0.09, -0.17, 0.13 }, {0.34, 0.22, -0.16, -0.0, -0.29, 0.15, -0.26, 0.59, -0.01, -0.05, 0.02, 0.05, 0.03, 0.06, -0.11, -0.63, 0.06, -0.13, 0.23, 0.3, -0.2, 0.42, -0.11, 0.01, -0.42, -0.26, -0.08, -0.07, -0.13, 0.17, -0.72, -0.42, -0.23, -0.26, 0.06, -0.21, 0.07, -0.32, -0.14, 0.12, 0.27, 0.29, -0.26, -0.48, -0.0, -0.04, -0.18, 0.04, 0.37, 0.03, -0.02, -0.12, -0.31, 0.13, -0.0, -0.17, -0.58, 0.17, -0.17, -0.01 }, {0.63, 0.33, -0.06, -0.1, -0.51, 0.25, -0.01, 0.74, 0.11, 0.01, 0.25, 0.01, -0.04, 0.31, -0.26, -0.03, 0.17, -0.17, 0.37, 0.33, -0.48, 0.06, -0.47, -0.08, -0.45, -0.37, 0.03, -0.09, -0.04, 0.65, -0.95, -0.19, 0.61, -0.1, 0.45, -0.58, -0.03, -0.3, -0.32, 0.15, 0.4, 0.31, -0.14, -1.07, -0.14, -0.05, -0.06, 0.31, 0.68, 0.02, -0.15, -0.52, -0.36, -0.12, -0.12, 0.17, -0.7, 0.39, -0.11, 0.23 }, {0.51, 0.42, -0.22, -0.06, -0.44, -0.21, 0.13, 0.22, 0.09, 0.02, 0.6, 0.06, 0.11, 0.19, -0.03, 0.15, 0.15, -0.0, 0.2, 0.26, -0.11, -0.24, -0.61, -0.14, -0.34, -0.21, -0.07, -0.48, -0.29, 0.7, -0.55, 0.6, 0.4, -0.15, 0.52, -0.23, 0.11, -0.1, -0.28, 0.09, 0.1, 0.2, 0.22, -0.4, 0.15, -0.14, 0.12, 0.04, 0.19, 0.06, -0.06, -0.39, -0.65, 0.19, -0.37, 0.06, -0.11, -0.16, 0.03, 0.36 }, {-0.01, -0.1, -0.44,

0.57, -0.07, 0.42, 0.12, 0.07, 0.27, 0.05, 0.1, 0.03, 0.13, 0.76, -0.63, 0.24, -0.03, -0.21, -0.26, 0.23, -0.64, -0.32, 0.13, -0.11, 0.15, -0.51, -0.22, -0.29, -0.03, 0.5, -0.24, -0.16, 0.11, -0.11, 0.45, -0.24, -0.02, -0.06, -0.59, -0.01, 0.65, 0.26, 0.27, -0.01, -0.16, -0.11, 0.29, 0.07, -0.05, 0.1, -0.62, -0.23, -0.14, 0.31, -0.31, -0.51, -0.27, 0.19, 0.1, -0.19 }, {0.3, 0.41, -0.69, 0.32, -0.12, 0.25, 0.41, 0.01, 0.25, 0.06, 0.3, 0.09, 0.11, 0.49, -0.2, 0.11, -0.0, 0.03, 0.27, 0.09, -0.65, -0.41, -0.41, 0.0, 0.11, -0.44, -0.28, -0.32, 0.23, 0.51, -0.45, 0.28, 0.03, 0.03, 0.59, -0.5, 0.08, -0.61, -0.93, -0.66, 0.4, 0.17, -0.04, -0.19, 0.07, -0.12, -0.02, 0.23, 0.1, 0.1, -0.59, 0.15, 0.17, 0.46, -0.6, 0.13, -0.29, 0.35, -0.12, 0.25 }, {0.15, 0.2, -0.31, 0.4, -0.19, 0.37, -0.15, -0.91, 0.21, 0.09, 0.43, 0.06, 0.18, 0.32, -0.03, 0.23, -0.05, -0.15, 0.21, -0.68, -0.28, -0.26, -0.3, -0.16, 0.11, -0.28, 0.01, 0.13, 0.32, 0.21, 0.02, 0.18, 0.23, 0.74, -0.01, 0.08, 0.01, 0.3, 0.05, 0.03, -0.11, 0.12, 0.07, 0.19, 0.2, 0.2, -0.08, 0.26, -0.1, 0.13, -0.27, 0.97, -0.14, 0.23, -0.19, 0.29, -0.01, 0.38, 0.31, -0.39 }, {0.07, 0.12, -0.22, 0.61, -0.28, 0.42, 0.13, -0.66, 0.19, 0.02, 0.55, 0.1, 0.06, 0.44, -0.48, 0.38, 0.01, 0.15, 0.13, -0.37, -0.31, -0.23, -0.67, -0.12, 0.1, -0.3, -0.16, -0.25, 0.45, -0.11, -0.16, -0.1, 0.3, 1.03, 0.07, 0.05, -0.02, 0.16, 0.04, 0.18, -0.37, -0.15, -0.1, 0.12, 0.03, 0.12, -0.18, 0.52, -0.01, 0.13, -0.52, 0.84, -0.28, 0.51, -0.23, -0.33, -0.08, 0.08, 0.33, -0.85 }, {0.04, 0.34, 0.17, 0.42, 0.0, 0.19, -0.15, -0.53, -0.03, -0.0, 0.55, 0.1, 0.09, 0.32, -0.36, -0.13, -0.11, -0.06, 0.21, -0.39, 0.13, 0.1, -0.25, -0.07, 0.21, -0.41, -0.16, -0.38, 0.5, -0.7, 0.13, -0.04, 0.13, 0.99, -0.14, 0.33, 0.19, 0.16, 0.02, 0.55, -0.37, 0.19, 0.46, 0.24, 0.23, 0.28, -0.26, 0.22, 0.03, 0.04, -0.33, 0.79, 0.05, 0.67, -0.08, -0.52, -0.35, -0.41, 0.37, -0.77 }, {0.05, 0.53, -0.16, 0.13, -0.08, -0.01, 0.0, -0.41, -0.19, 0.04, 0.64, 0.12, -0.08, 0.29, -0.23, 0.38, -0.41, -0.19, 0.33, -0.12, -0.03, -0.5, -0.72, 0.0, 0.29, -0.14, -0.21, -0.4, 0.79, -0.16, -0.21, 0.33, 0.08, 1.01, 0.16, 0.1, -0.06, 0.06, -0.05, 0.39, -0.28, -0.09, 0.23, -0.14, 0.18, 0.07, -0.34, 0.45, 0.01, -0.05, -0.37, 0.55, -0.0, 0.26, -0.32, -0.2, 0.35, -0.21, 0.15, -0.6 }, {-0.4, -0.31, 0.28, 0.96, -0.11, 0.65, 0.13, -0.68, -0.05, 0.1, -0.06, 0.04, -0.0, 0.08, -

0.09, 0.2, -0.16, -0.19, -0.39, -0.41, -0.17, -0.31, 0.48, 0.22, 0.02, 0.04, 0.17, -0.03, -0.01, -0.2, 0.17, 0.0, 0.24, 0.95, 0.19, 0.62, 0.07, -0.25, -0.29, -0.23, -0.39, -0.03, 0.46, 0.01, 0.26, 0.09, 0.32, -0.32, -0.48, 0.07, -0.31, 0.74, 0.91, 0.14, -0.44, -0.13, -0.19, -0.72, 0.41, -0.7 }, {-0.22, -0.49, 0.26, 0.66, -0.22, 0.52, 0.2, -0.33, 0.27, 0.12, 0.09, 0.06, 0.07, -0.21, 0.21, 0.36, -0.03, -0.35, -0.73, -0.25, -0.03, -0.27, 0.29, -0.15, -0.01, 0.03, -0.18, -0.27, -0.01, -0.12, 0.06, -0.21, -0.03, 0.18, 0.21, 0.28, 0.07, -0.03, -0.37, -0.05, -0.06, -0.11, 0.31, -0.15, 0.0, 0.1, 0.53, 0.0, -0.56, 0.08, -0.18, -0.09, 0.41, -0.1, -0.25, -0.01, -0.1, -0.37, 0.18, -0.6 }, {-0.27, -0.33, -0.13, 0.02, 0.28, 0.08, 0.12, 0.24, -0.06, -0.07, -0.09, 0.0, -0.1, -0.29, 0.05, -0.12, -0.09, -0.2, -0.42, 0.12, -0.03, -0.03, 0.12, -0.06, 0.01, 0.14, 0.11, 0.04, -0.01, 0.32, -0.17, 0.02, -0.0, -0.46, -0.01, 0.19, -0.09, -0.19, 0.12, 0.17, -0.1, 0.0, -0.36, -0.39, -0.03, -0.16, 0.25, -0.13, -0.14, 0.01, -0.01, -0.54, 0.06, -0.04, 0.19, -0.2, -0.04, -0.02, -0.46, -0.14 }, {-0.45, 0.17, -0.09, -0.09, 0.73, -0.01, -0.02, 0.19, -0.05, 0.03, -0.02, 0.06, -0.06, -0.14, 0.38, -0.23, -0.19, 0.06, -0.06, 0.35, 0.12, -0.19, 0.12, 0.08, -0.04, 0.24, 0.11, 0.08, 0.14, 0.3, 0.07, 0.13, -0.12, -0.46, -0.08, 0.43, -0.0, 0.16, 0.27, 0.03, 0.04, 0.16, -0.41, -0.19, 0.16, -0.15, 0.05, -0.18, 0.07, -0.1, 0.23, -0.09, 0.07, -0.02, 0.35, 0.21, 0.23, 0.23, -0.44, -0.09 }, {-0.63, -0.22, 0.04, 0.07, 0.54, 0.07, -0.08, 0.07, 0.07, 0.03, -0.34, 0.02, -0.03, -0.41, 0.25, -0.18, 0.2, 0.04, -0.5, 0.41, 0.69, -0.03, 0.39, -0.15, 0.05, 0.09, 0.19, 0.45, 0.28, -0.13, -0.1, -0.08, -0.07, -0.58, -0.26, 0.74, 0.03, 0.08, 0.01, 0.03, -0.26, 0.09, -0.39, 0.02, 0.13, 0.07, 0.34, -0.47, -0.15, 0.06, 0.84, -0.26, -0.06, -0.05, 0.47, -0.04, -0.11, 0.03, -0.5, 0.1 }, {-0.74, -0.39, -0.01, -0.36, 0.58, -0.04, -0.19, -0.05, 0.07, 0.03, -0.31, 0.09, 0.09, -0.74, 0.48, -0.47, -0.02, -0.1, -0.7, 0.23, 0.75, 0.45, 0.44, -0.05, -0.12, 0.05, 0.12, 0.61, 0.09, -0.19, -0.03, -0.11, 0.05, -0.75, -0.34, 0.84, 0.05, 0.26, 0.53, 0.16, -0.21, 0.14, -0.25, 0.19, 0.23, 0.13, 0.41, -0.52, -0.46, 0.05, 1.04, -0.37, 0.08, 0.12, 0.6, -0.11, -0.42, 0.28, -0.58, 0.46 }, {-0.18, 0.12, -0.05, 0.08, 0.08, -0.28, -0.2, 0.3, 0.0, 0.01, -0.06, 0.1, -0.03, 0.21, 0.08, -0.23, 0.14, 0.08, 0.13, 0.02, 0.15,

-0.2, -0.17, -0.3, -0.3, -0.04, -0.14, -0.03, -0.26, -0.05, -0.12, -0.01, -0.19, -0.16, 0.18, 0.06, -0.02, -0.09, 0.13, -0.22, -0.08, 0.23, -0.2, -0.43, -0.15, 0.05, -0.18, -0.15, 0.19, -0.04, 0.25, 0.03, -0.44, -0.05, -0.24, 0.06, 0.03, 0.2, -0.24, 0.1 }, {0.03, 0.28, -0.01, -0.03, 0.05, -0.27, -0.2, 0.52, -0.01, -0.05, 0.07, 0.09, 0.02, 0.19, 0.04, -0.19, 0.11, -0.0, 0.28, -0.05, 0.11, -0.33, -0.12, -0.34, -0.38, 0.06, -0.05, -0.09, -0.15, 0.04, -0.2, -0.12, -0.29, -0.1, 0.17, -0.08, -0.07, -0.07, 0.15, -0.19, 0.04, 0.33, -0.22, -0.45, -0.17, 0.03, -0.3, -0.09, 0.33, -0.05, 0.08, 0.03, -0.35, -0.21, -0.22, 0.25, -0.01, 0.39, -0.25, 0.14 }, {0.07, 0.23, -0.16, 0.34, -0.15, 0.19, -0.13, 0.55, -0.02, 0.01, 0.16, 0.04, -0.02, 0.31, -0.3, 0.08, 0.06, 0.24, 0.12, 0.07, -0.03, -0.21, -0.11, -0.17, -0.33, -0.46, -0.04, -0.16, -0.02, 0.24, -0.03, 0.07, 0.28, -0.18, 0.1, -0.13, 0.02, -0.18, -0.27, 0.15, 0.07, 0.28, 0.29, -0.36, -0.03, 0.02, -0.24, -0.01, 0.28, 0.03, -0.06, 0.01, -0.28, -0.06, -0.22, 0.07, -0.17, 0.28, -0.07, 0.04 }, {-0.0, 0.09, -0.33, -0.01, -0.29, 0.08, 0.09, 0.13, 0.19, 0.05, 0.34, 0.05, 0.09, -0.22, -0.37, 0.04, 0.0, 0.46, -0.04, 0.02, -0.19, -0.03, -0.44, -0.14, -0.2, -0.47, -0.15, -0.4, 0.03, 0.31, 0.26, 0.37, 0.3, -0.12, 0.13, -0.12, 0.13, -0.13, -0.63, 0.22, -0.05, 0.21, 0.56, 0.09, 0.35, 0.1, 0.23, -0.06, -0.44, 0.15, 0.2, -0.04, -0.39, 0.44, -0.21, -0.43, 0.16, -0.18, 0.13, 0.22 }, {0.26, 0.24, -0.4, 0.13, -0.49, 0.48, 0.26, -0.13, 0.3, 0.07, 0.39, 0.05, 0.11, 0.18, -0.28, 0.09, -0.37, 0.01, -0.07, 0.31, -0.8, -0.06, -0.13, 0.2, 0.06, -0.21, -0.23, -0.23, -0.11, 0.32, 0.05, -0.09, 0.15, 0.14, 0.46, -0.41, -0.06, -0.15, -0.63, 0.25, 0.52, 0.29, 0.45, -0.2, -0.11, 0.17, 0.15, 0.11, -0.21, 0.16, -0.11, -0.12, -0.31, 0.2, -0.17, -0.42, -0.33, -0.04, 0.24, -0.0 }, {0.53, 0.63, -0.59, -0.18, -0.46, 0.18, 0.33, -0.04, 0.08, 0.1, 0.53, 0.11, 0.12, -0.13, -0.04, -0.11, -0.21, 0.3, 0.35, 0.2, -0.82, -0.08, -0.47, 0.24, -0.26, -0.34, -0.31, -0.25, -0.18, 0.37, -0.16, 0.28, 0.16, -0.05, 0.35, -0.59, -0.03, -0.62, -0.87, -0.49, 0.22, 0.29, 0.14, -0.21, -0.02, 0.2, -0.28, 0.01, -0.25, 0.21, 0.25, 0.05, -0.1, 0.29, -0.46, 0.06, -0.6, -0.04, 0.11, 0.53 }, {0.3, 0.05, -0.22, 0.0, -0.43, 0.68, 0.02, -0.77, -0.05, 0.15, 0.38, 0.04, 0.17, -0.0, -0.03, 0.08, -0.44, -0.0, -0.14, -0.03, -0.74, -0.1, -0.12, 0.16, -0.27, -0.27, -0.28,

0.01, 0.13, 0.54, 0.06, 0.36, 0.22, 0.46, 0.31, -0.1, -0.01, -0.12, 0.14, 0.12, 0.48, 0.19, 0.2, -0.14, 0.17, 0.28, 0.3, 0.36, -0.29, 0.18, -0.36, 0.5, 0.21, 0.04, -0.41, 0.07, -0.57, 0.11, 0.37, -0.25 }, {0.47, -0.35, -0.19, 0.36, -0.78, 0.83, 0.39, -0.68, 0.07, 0.0, 0.31, 0.11, 0.12, 0.03, -0.27, 0.15, -0.39, -0.05, -0.37, 0.21, -0.53, -0.03, -0.4, 0.31, -0.24, -0.14, -0.34, -0.19, 0.15, -0.1, -0.44, -0.11, 0.1, 0.99, 0.45, -0.08, -0.15, -0.17, -0.11, 0.01, -0.15, -0.19, 0.09, -0.26, 0.02, 0.1, 0.49, 0.33, -0.45, 0.04, -0.38, 0.59, -0.05, 0.43, -0.34, -0.49, -0.38, -0.24, 0.42, -0.99 }, {-0.07, 0.21, -0.01, 0.66, 0.07, 0.46, -0.27, 0.15, -0.07, 0.11, 0.29, 0.12, 0.01, 0.3, -0.15, -0.34, 0.38, -0.3, -0.22, 0.15, 0.14, 0.33, -0.11, -0.01, -0.01, -0.7, 0.03, -0.29, 0.27, -0.22, -0.15, -0.3, 0.2, 0.36, 0.07, 0.25, 0.3, -0.25, -0.18, 0.32, -0.48, 0.24, 0.12, -0.35, 0.21, 0.16, -0.11, 0.03, -0.36, 0.05, -0.25, 0.6, -0.38, 0.82, 0.06, -0.67, -0.63, -0.43, 0.04, -0.7 }, {-0.21, 0.12, -0.04, 0.49, 0.25, 0.23, -0.13, 0.19, -0.17, 0.05, 0.27, 0.02, -0.07, 0.22, 0.12, -0.13, -0.15, -0.46, -0.06, 0.61, -0.1, -0.3, -0.36, 0.1, 0.21, -0.35, -0.05, -0.23, 0.51, 0.32, -0.2, -0.04, 0.26, 0.41, 0.23, 0.26, 0.17, -0.3, -0.13, -0.09, -0.13, -0.08, -0.15, -0.41, 0.16, -0.27, -0.04, 0.36, -0.02, 0.02, -0.54, 0.39, -0.2, 0.31, -0.12, -0.23, -0.27, -0.16, -0.09, -0.66 }, {-0.8, -0.15, 0.0, 0.6, 0.27, 0.13, -0.07, -0.56, 0.1, 0.1, -0.23, 0.09, 0.12, 0.08, 0.07, 0.03, -0.18, -0.23, -0.57, 0.19, 0.14, 0.08, 0.33, 0.16, 0.54, 0.04, 0.17, -0.07, 0.32, -0.04, 0.55, -0.09, 0.26, 0.56, -0.02, 0.62, 0.15, 0.15, 0.47, -0.05, -0.6, 0.02, 0.04, 0.38, 0.04, -0.08, 0.16, 0.14, -0.4, 0.14, -0.53, 0.58, 0.3, 0.2, -0.06, -0.21, 0.44, -0.34, 0.32, -0.58 }, {-0.2, -0.01, 0.25, 0.13, 0.03, 0.29, 0.15, -0.1, 0.18, 0.03, 0.07, 0.07, 0.01, -0.58, 0.29, 0.01, 0.11, -0.06, -0.41, -0.18, -0.09, 0.32, 0.01, -0.11, -0.21, -0.08, 0.28, 0.09, 0.04, -0.73, -0.1, -0.38, -0.1, 0.43, -0.05, 0.14, 0.1, 0.05, -0.33, -0.01, -0.41, -0.11, 0.27, -0.27, -0.05, 0.06, 0.04, 0.23, -0.39, 0.05, -0.03, 0.41, 0.39, 0.33, 0.03, -0.18, -0.02, -0.27, 0.09, -0.15 }, {0.01, -0.34, 0.1, -0.15, -0.09, 0.55, 0.25, 0.15, 0.05, -0.05, -0.18, 0.11, 0.0, -0.65, -0.06, -0.05, -0.21, -0.03, -0.4, 0.03, -0.25, 0.18, -0.16, 0.16, -0.02, 0.08, 0.37, 0.29, 0.1, -0.12, 0.02, -0.02, 0.0, 0.18, 0.1, 0.19, -0.04,

-0.11, -0.35, 0.13, -0.37, 0.04, 0.1, -0.55, -0.08, -0.1, 0.26, -0.09, -0.31, 0.03, -0.04, -0.01, 0.17, 0.47, 0.21, -0.39, -0.05, -0.31, -0.17, -0.13 }, {-0.16, 0.11, -0.01, -0.37, 0.25, 0.49, -0.17, 0.14, 0.05, 0.04, -0.07, 0.11, 0.03, -0.78, 0.14, -0.51, -0.37, 0.17, -0.04, 0.15, -0.22, 0.25, -0.04, 0.13, -0.25, 0.01, 0.3, 0.29, -0.06, -0.33, 0.16, 0.27, -0.18, -0.18, -0.32, 0.02, -0.08, 0.07, -0.19, 0.15, -0.08, 0.11, -0.1, -0.3, -0.05, -0.16, -0.0, -0.18, -0.06, -0.08, 0.4, 0.14, 0.08, 0.54, 0.4, -0.36, 0.03, -0.24, -0.3, 0.05 }, {-0.11, -0.13, 0.03, -0.25, 0.01, -0.1, -0.02, 0.13, 0.04, 0.09, -0.08, 0.05, 0.03, -0.42, 0.11, -0.16, -0.11, 0.43, -0.29, 0.23, 0.16, -0.07, 0.33, -0.12, -0.17, 0.12, 0.43, 0.28, 0.26, 0.16, -0.31, 0.07, 0.03, -0.34, -0.17, 0.08, 0.09, 0.09, -0.17, -0.03, 0.1, 0.05, -0.19, -0.36, 0.15, -0.09, 0.36, -0.32, -0.12, -0.0, 0.34, -0.1, 0.02, 0.12, 0.21, -0.05, -0.06, -0.17, -0.28, 0.0 }, {-0.61, -0.46, 0.02, -0.21, 0.24, -0.33, -0.1, -0.3, -0.03, 0.13, -0.23, 0.08, -0.01, -0.72, 0.26, -0.09, -0.13, 0.37, -0.48, -0.02, 0.54, -0.0, 0.83, -0.2, -0.17, 0.16, 0.47, 0.57, 0.25, 0.34, -0.01, 0.05, 0.01, -0.31, 0.01, 0.45, 0.1, 0.26, 0.31, -0.28, -0.12, 0.08, -0.12, -0.22, 0.21, -0.12, 0.41, -0.61, -0.45, 0.0, 0.59, -0.38, 0.07, 0.01, 0.11, 0.12, -0.28, 0.15, -0.31, 0.1 }, {-0.22, 0.46, -0.22, 0.06, 0.29, -0.48, -0.24, 0.15, -0.09, -0.05, 0.05, -0.0, 0.01, 0.39, 0.23, -0.23, -0.04, -0.5, 0.37, -0.16, -0.08, -0.19, -0.12, -0.1, -0.11, 0.22, -0.29, 0.12, -0.03, -0.31, 0.2, 0.25, -0.14, -0.12, 0.21, 0.2, -0.11, 0.25, 0.17, 0.06, 0.13, 0.21, -0.41, -0.06, -0.17, -0.18, -0.42, 0.1, 0.39, -0.08, -0.02, -0.07, -0.37, -0.25, -0.33, 0.02, 0.14, 0.61, -0.14, 0.22 }, {-0.14, 0.42, -0.23, 0.02, 0.24, -0.48, -0.18, 0.2, -0.05, -0.04, 0.15, 0.1, -0.02, 0.37, 0.23, -0.01, -0.1, -0.39, 0.39, -0.23, -0.06, -0.11, -0.09, -0.22, -0.02, 0.17, -0.29, 0.01, 0.07, -0.25, 0.3, 0.29, -0.18, -0.17, 0.14, 0.15, -0.05, 0.35, 0.22, 0.12, 0.1, 0.23, -0.26, -0.02, -0.24, -0.14, -0.39, 0.15, 0.43, -0.06, -0.12, -0.02, -0.36, -0.2, -0.38, 0.05, 0.22, 0.58, -0.15, 0.09 }, {0.07, 0.28, -0.16, 0.24, -0.22, 0.13, 0.05, 0.23, 0.04, 0.07, 0.34, 0.09, 0.05, 0.42, -0.16, 0.03, -0.1, -0.01, 0.08, 0.01, -0.17, -0.15, 0.16, -0.1, -0.25, -0.26, -0.16, 0.01, 0.11, 0.0, 0.02, 0.38, 0.28, -0.16, 0.06, -0.04, 0.02, -0.12, -0.37, 0.06, 0.15,

0.25, 0.3, -0.2, 0.07, 0.06, -0.16, 0.05, 0.08, 0.06, -0.17, 0.13, -0.16, 0.19, -0.27, -0.04, -0.01, 0.22, 0.06, 0.1 }, {0.27, 0.35, -0.14, 0.21, -0.11, -0.06, 0.23, 0.19, 0.16, 0.03, 0.27, 0.1, 0.04, 0.27, -0.02, -0.09, 0.06, 0.07, 0.06, 0.16, -0.25, -0.2, -0.06, -0.01, -0.48, -0.11, -0.17, -0.26, -0.23, 0.24, -0.14, 0.33, 0.17, -0.07, 0.2, -0.29, 0.11, -0.03, -0.71, -0.03, -0.14, 0.27, 0.17, -0.04, 0.04, 0.26, 0.15, -0.08, -0.24, 0.08, -0.01, 0.02, -0.32, 0.12, -0.37, -0.09, 0.06, -0.04, 0.09, -0.0 }, {0.57, 0.25, -0.15, -0.13, -0.23, 0.15, 0.06, -0.2, 0.31, 0.06, 0.44, 0.02, 0.14, 0.33, 0.04, 0.1, -0.01, 0.12, 0.21, -0.16, -0.66, -0.21, -0.36, 0.02, -0.35, 0.19, -0.18, -0.12, -0.47, 0.58, -0.09, 0.08, 0.11, -0.05, 0.44, -0.53, 0.03, 0.18, -0.25, 0.05, 0.18, 0.26, 0.16, -0.15, -0.34, 0.34, -0.16, -0.03, 0.07, 0.17, 0.03, -0.35, -0.36, -0.24, -0.36, 0.11, -0.2, 0.05, 0.1, 0.3 }, {0.79, 0.96, -0.51, -0.37, -0.08, -0.21, 0.11, -0.03, 0.07, -0.0, 0.85, 0.11, 0.11, 0.19, 0.17, 0.09, 0.03, 0.47, 0.79, -0.28, -0.71, -0.18, -0.62, 0.13, -0.31, 0.14, -0.09, -0.24, -0.21, 0.32, -0.17, 0.26, -0.16, -0.09, 0.25, -0.56, -0.07, -0.17, -0.49, -0.26, -0.06, 0.31, -0.12, -0.2, -0.05, 0.15, -0.65, 0.1, 0.21, 0.06, 0.2, -0.04, -0.22, -0.04, -0.6, 0.44, -0.3, -0.01, 0.06, 0.54 }, {0.42, -0.01, -0.34, -0.0, -0.15, 0.05, 0.23, -0.71, -0.19, 0.05, 0.8, 0.03, 0.05, 0.21, 0.44, 0.08, -0.33, 0.12, 0.18, -0.38, -0.95, -0.27, -0.1, 0.02, -0.47, -0.08, -0.45, -0.02, -0.04, 0.4, 0.29, 0.54, 0.05, 0.3, 0.33, -0.24, -0.15, 0.04, 0.42, -0.02, 0.43, 0.14, -0.14, 0.04, 0.29, -0.07, 0.05, 0.15, -0.1, 0.06, 0.08, 0.16, 0.17, -0.1, -0.43, 0.36, 0.35, 0.08, 0.26, 0.09 }, {0.35, -0.55, -0.05, 0.49, -0.47, 0.5, 0.53, -0.73, 0.06, 0.13, -0.03, 0.13, 0.09, -0.16, 0.26, 0.1, -0.47, 0.07, -0.41, 0.05, -0.92, -0.03, 0.32, 0.28, -0.81, -0.38, -0.36, 0.06, -0.34, -0.65, -0.24, -0.04, 0.35, 0.74, 0.41, -0.03, -0.07, -0.28, 0.01, -0.51, -0.07, -0.14, -0.15, -0.1, 0.1, -0.1, 0.62, -0.08, -0.22, 0.1, 0.1, 0.05, 0.47, 0.5, -0.42, -0.24, 0.03, -0.54, 0.41, -0.47 }, {-0.34, 0.33, -0.01, 0.5, 0.3, 0.27, -0.1, -0.01, 0.02, 0.11, -0.06, 0.13, 0.16, 0.3, 0.1, -0.27, 0.16, -0.24, -0.05, 0.22, -0.1, 0.19, 0.01, 0.16, -0.09, -0.44, -0.17, -0.17, 0.09, -0.26, -0.52, -0.39, 0.44, -0.04, 0.24, 0.15, 0.32, -0.3, 0.1, 0.02, -0.37, 0.06, -0.05, -0.31, 0.26, 0.04, 0.05, -0.0, -0.05, 0.21, -0.24, -0.09, 0.09,

0.56, -0.11, -0.33, -0.36, -0.27, 0.04, -0.55 }, {-0.32, 0.31, -0.22, 0.16, 0.36, 0.15, -0.06, 0.3, 0.04, 0.06, 0.02, 0.09, 0.12, 0.22, 0.16, -0.08, -0.04, -0.53, 0.2, 0.54, -0.38, -0.22, -0.51, 0.14, 0.22, -0.11, -0.12, -0.16, 0.39, 0.16, -0.56, -0.24, 0.43, 0.22, 0.49, 0.01, 0.13, -0.24, 0.03, -0.06, -0.1, -0.15, -0.07, -0.33, 0.13, -0.22, 0.01, 0.56, 0.29, 0.13, -0.48, 0.04, 0.08, 0.13, -0.15, -0.06, -0.3, -0.14, -0.03, -0.73 }, {-0.63, -0.19, 0.05, 0.54, 0.16, 0.62, 0.16, -0.46, 0.2, 0.08, -0.33, -0.0, 0.06, -0.07, -0.02, -0.03, -0.05, -0.39, -0.33, 0.08, -0.12, 0.04, -0.23, 0.25, 0.43, 0.12, 0.07, -0.2, 0.13, -0.26, 0.1, -0.2, 0.33, 0.41, 0.37, 0.45, 0.11, 0.14, 0.57, 0.07, -0.77, -0.2, -0.1, 0.11, -0.04, -0.13, 0.23, 0.53, -0.04, 0.06, -0.65, 0.34, -0.25, 0.17, -0.14, -0.18, 0.46, -0.57, 0.06, -0.63 }, {-0.5, 0.05, 0.15, 0.17, 0.17, 0.38, 0.27, 0.02, 0.19, 0.02, -0.11, 0.07, 0.15, -0.21, 0.24, -0.08, 0.28, 0.04, 0.01, 0.14, -0.02, 0.09, -0.13, -0.22, -0.07, -0.22, 0.42, -0.12, 0.17, -0.44, -0.03, -0.47, -0.09, 0.1, 0.11, 0.27, 0.13, 0.23, -0.3, -0.08, -0.39, -0.21, -0.08, -0.23, -0.12, 0.0, -0.16, 0.39, -0.23, 0.1, -0.15, -0.07, -0.07, 0.32, -0.17, -0.03, -0.02, -0.27, 0.04, -0.17 }, {-0.01, -0.21, 0.04, -0.55, -0.04, 0.24, 0.17, -0.05, 0.01, -0.01, -0.21, 0.01, -0.01, -0.46, 0.1, 0.18, -0.25, 0.1, -0.1, -0.07, -0.03, 0.06, -0.13, -0.1, 0.22, 0.12, 0.47, 0.07, 0.08, 0.06, 0.14, 0.11, -0.04, 0.11, 0.18, -0.18, -0.08, 0.22, -0.27, 0.13, -0.04, -0.13, -0.03, -0.57, -0.14, -0.27, 0.12, 0.11, -0.13, -0.09, 0.04, -0.25, -0.38, 0.21, 0.09, -0.22, 0.22, -0.19, -0.12, -0.25 }, {-0.15, 0.16, -0.22, -0.8, 0.23, 0.28, -0.21, 0.18, -0.07, -0.07, -0.07, 0.05, 0.03, -0.69, 0.25, -0.71, -0.42, 0.28, 0.16, 0.26, 0.09, 0.15, -0.05, -0.11, -0.23, -0.11, 0.34, 0.2, 0.01, 0.2, 0.02, 0.48, -0.2, -0.56, -0.24, -0.21, -0.01, 0.09, -0.2, 0.36, 0.4, 0.16, -0.16, -0.4, -0.12, -0.29, -0.08, -0.16, 0.04, -0.11, 0.49, -0.15, -0.33, 0.4, 0.38, -0.24, 0.29, -0.0, -0.52, 0.29 }, {0.07, -0.14, -0.12, -0.67, -0.11, 0.05, -0.12, -0.03, 0.15, 0.1, -0.24, 0.11, 0.05, -0.44, 0.28, 0.03, -0.37, 0.05, -0.13, 0.06, 0.13, -0.05, 0.12, -0.08, -0.0, -0.15, 0.29, 0.24, 0.28, 0.08, -0.59, 0.15, 0.24, -0.09, -0.06, -0.24, 0.03, 0.23, -0.18, 0.1, 0.12, 0.01, -0.13, -0.41, 0.0, -0.03, 0.26, -0.12, -0.25, 0.06, 0.4, 0.12, -0.05, 0.22, 0.31, -0.35, 0.19, -0.42, -0.36,

0.27 }, {-0.28, 0.09, -0.02, -0.51, 0.25, -0.32, -0.08, -0.24, -0.02, 0.12, -0.22, 0.08, 0.01, -0.25, 0.46, 0.25, -0.25, -0.01, -0.05, -0.23, 0.19, -0.18, 0.64, -0.32, 0.16, 0.08, 0.27, 0.38, 0.18, 0.18, -0.36, -0.05, 0.04, -0.02, 0.01, -0.15, 0.02, 0.37, 0.22, -0.18, 0.19, 0.06, -0.14, -0.32, 0.1, -0.08, 0.15, -0.43, -0.45, 0.09, 0.52, -0.15, -0.04, -0.06, 0.21, 0.06, -0.17, 0.27, -0.19, -0.05 }, {-0.16, 0.07, -0.23, -0.29, 0.28, -0.38, -0.03, 0.05, 0.19, 0.13, -0.01, 0.1, 0.13, -0.03, 0.23, 0.48, 0.0, -0.15, 0.22, -0.29, -0.48, -0.24, -0.43, 0.04, -0.06, 0.2, -0.46, 0.23, -0.08, 0.13, -0.1, 0.18, 0.28, 0.09, 0.44, -0.14, 0.05, 0.21, 0.4, 0.04, 0.17, 0.16, -0.23, 0.31, -0.1, -0.05, -0.14, 0.12, 0.34, 0.15, 0.07, -0.22, -0.25, -0.47, -0.41, 0.19, 0.3, 0.41, -0.0, 0.15 }, {-0.04, 0.19, -0.37, -0.45, 0.27, -0.38, -0.17, 0.02, 0.21, 0.1, 0.15, 0.13, 0.05, -0.02, 0.2, 0.53, -0.08, -0.11, 0.21, -0.34, -0.39, -0.15, -0.51, -0.02, 0.12, 0.29, -0.45, 0.15, 0.11, 0.16, -0.03, 0.32, 0.33, 0.12, 0.42, -0.28, 0.0, 0.4, 0.34, 0.12, 0.19, 0.25, -0.12, 0.26, -0.16, -0.12, -0.29, 0.22, 0.39, 0.14, 0.15, -0.2, -0.31, -0.49, -0.32, 0.27, 0.27, 0.58, 0.1, 0.3 }, {-0.01, 0.38, -0.14, -0.23, -0.03, -0.47, -0.25, -0.12, 0.12, 0.06, 0.27, 0.12, 0.11, 0.16, 0.18, 0.4, -0.09, 0.55, 0.18, -0.16, -0.19, -0.2, -0.01, -0.25, -0.03, 0.13, -0.45, 0.06, 0.11, 0.42, 0.19, 0.2, -0.11, -0.3, 0.2, 0.08, 0.15, 0.2, 0.18, 0.02, 0.27, 0.27, 0.06, 0.42, 0.2, 0.12, -0.22, -0.14, 0.03, 0.08, -0.07, -0.29, -0.31, -0.02, -0.5, 0.11, 0.17, 0.26, 0.15, 0.26 }, {0.03, -0.33, 0.01, -0.09, -0.09, -0.01, -0.17, -0.16, 0.2, 0.02, -0.08, 0.1, -0.01, -0.12, -0.32, 0.24, -0.32, 0.02, -0.21, -0.05, -0.19, -0.24, 0.04, -0.16, -0.06, -0.18, 0.06, 0.3, -0.22, 0.38, 0.42, 0.67, 0.31, -0.29, 0.18, 0.1, 0.08, -0.14, -0.12, -0.08, 0.12, 0.27, 0.16, 0.19, 0.07, 0.2, 0.15, -0.28, -0.52, 0.06, 0.05, -0.44, -0.33, 0.06, -0.25, 0.1, 0.6, 0.01, 0.35, 0.24 }, {0.34, -0.71, -0.07, -0.15, -0.37, 0.36, -0.36, -0.41, 0.09, -0.02, -0.08, 0.07, 0.04, 0.1, -0.24, -0.59, -0.33, -0.05, -0.26, 0.36, -0.26, 0.07, 0.05, -0.13, -0.39, -0.44, 0.16, 0.18, -0.37, 0.2, -0.02, 0.24, 0.2, -0.43, -0.1, -0.3, -0.07, -0.2, -0.33, 0.31, 0.39, 0.27, 0.52, 0.25, -0.07, 0.31, 0.44, -0.32, -0.45, -0.0, 0.23, -0.59, -0.29, 0.37, 0.02, -0.32, 0.08, -0.02, 0.4, 0.4 }, {0.3, -0.26, -0.14, -0.04, -0.1, 0.03,

-0.25, -0.28, -0.05, 0.01, 0.45, 0.01, 0.06, -0.14, 0.01, -0.07, -0.17, 0.15, 0.06, 0.09, -0.39, -0.31, -0.1, -0.15, -0.11, -0.19, 0.06, 0.1, -0.08, 0.13, 0.12, 0.49, -0.07, -0.11, -0.04, -0.43, -0.12, -0.18, -0.39, -0.14, 0.31, 0.33, 0.28, 0.23, 0.29, 0.16, -0.0, -0.24, -0.43, -0.0, 0.08, -0.46, -0.19, 0.23, -0.23, 0.18, 0.34, 0.03, 0.35, 0.33 }, {0.5, -0.62, -0.1, -0.03, -0.21, 0.06, 0.1, -0.66, -0.38, -0.02, 0.78, 0.1, -0.02, -0.02, 0.55, 0.09, -0.6, 0.34, -0.28, -0.29, -0.38, -0.42, 0.9, -0.05, -0.87, 0.14, -0.29, 0.4, -0.05, 0.34, 0.08, 0.91, 0.13, -0.13, -0.1, -0.21, -0.08, 0.17, 0.07, -0.65, 0.66, 0.14, -0.1, -0.07, 0.6, 0.0, 0.66, -0.42, -0.51, -0.07, 0.61, -0.22, 0.19, -0.53, -0.53, 0.9, 0.42, 0.41, 0.29, 0.58 }, {0.52, -0.77, -0.06, -0.01, -0.48, -0.06, 0.18, -0.64, -0.08, -0.02, 0.22, -0.01, -0.0, -0.21, 0.23, 0.05, -0.84, 0.18, -0.4, -0.1, -0.6, -0.44, 0.65, -0.02, -0.7, 0.26, -0.56, 0.28, -0.05, -0.35, -0.36, 0.42, 0.48, 0.1, 0.29, -0.37, -0.24, -0.04, 0.32, -0.44, 0.64, -0.14, -0.12, -0.26, 0.28, -0.18, 0.86, -0.23, -0.11, 0.0, 0.27, -0.45, 0.57, -0.59, -0.57, 0.27, 0.64, 0.44, 0.32, 0.08 }, {0.2, 0.32, -0.16, -0.47, -0.11, -0.45, -0.29, 0.11, -0.05, -0.04, 0.16, -0.0, -0.08, -0.2, -0.09, 0.44, 0.03, 0.23, -0.13, -0.34, -0.37, -0.17, -0.1, -0.38, 0.32, -0.13, -0.36, 0.13, 0.22, -0.04, 0.3, 0.3, 0.52, 0.07, 0.1, -0.33, 0.02, 0.27, 0.45, 0.08, 0.07, 0.07, 0.27, 0.19, 0.17, -0.01, -0.33, 0.14, 0.25, 0.03, -0.24, -0.27, 0.09, -0.52, -0.41, -0.11, 0.31, 0.44, 0.23, -0.06 }, {0.06, 0.34, -0.19, -0.5, 0.19, -0.18, -0.09, 0.23, -0.07, -0.03, 0.14, -0.01, -0.1, -0.29, 0.09, 0.45, -0.1, -0.24, 0.04, -0.02, -0.5, -0.04, -0.27, -0.26, 0.35, 0.11, -0.26, 0.16, 0.53, -0.03, 0.04, 0.39, 0.39, 0.42, 0.31, -0.32, 0.02, 0.16, 0.19, 0.12, -0.11, -0.09, 0.04, -0.14, 0.13, -0.15, -0.09, 0.57, 0.66, -0.0, -0.52, -0.07, 0.25, -0.69, -0.29, -0.26, 0.11, 0.45, 0.06, -0.21 }, {-0.26, -0.49, 0.06, 0.21, -0.12, 0.57, -0.02, -0.5, 0.08, -0.03, -0.49, 0.1, 0.05, -0.2, -0.4, 0.87, 0.08, -0.39, -0.26, -0.27, -0.51, -0.11, -0.07, 0.1, 0.56, 0.33, -0.05, -0.26, 0.1, -0.16, 0.13, -0.11, 0.21, 1.01, 0.26, -0.25, 0.11, 0.3, 0.52, 0.2, -0.25, -0.15, 0.39, 0.02, -0.12, -0.08, 0.08, 0.46, -0.23, -0.01, -0.87, 0.34, -0.11, -0.21, -0.24, -0.38, 0.42, -0.3, 0.3, -0.75 }, {-0.5, 0.03, 0.11, -0.09, 0.28, 0.43, -0.05, -0.06, 0.17, 0.01, -0.47, -0.01, -0.04, -0.32,

-0.03, 0.42, 0.23, 0.12, 0.05, -0.04, 0.01, 0.04, 0.0, -0.18, 0.25, 0.27, 0.43, -0.02, 0.16, -0.35, 0.33, -0.37, -0.06, 0.38, 0.19, -0.03, 0.09, 0.46, 0.2, -0.2, -0.29, -0.16, 0.04, -0.05, -0.21, 0.04, -0.37, 0.41, -0.1, 0.02, -0.4, 0.29, -0.2, -0.03, -0.13, -0.12, 0.52, -0.2, 0.1, -0.61 }, {-0.39, -0.01, 0.02, -0.04, 0.17, 0.41, 0.18, -0.34, 0.21, 0.12, -0.4, 0.09, 0.01, -0.23, -0.01, 0.26, 0.02, 0.12, -0.17, 0.07, -0.1, 0.06, 0.06, 0.04, 0.37, 0.31, 0.15, -0.2, 0.04, 0.08, -0.08, -0.06, 0.02, 0.3, 0.14, 0.11, 0.13, 0.4, -0.03, 0.1, -0.1, -0.11, -0.08, -0.19, -0.33, 0.03, -0.04, 0.16, -0.21, 0.06, -0.35, 0.1, -0.33, 0.25, 0.07, -0.29, 0.31, -0.41, 0.17, -0.49 }, {-0.2, 0.22, -0.02, -0.3, 0.36, 0.1, -0.04, -0.02, -0.02, 0.11, -0.12, 0.08, -0.03, -0.33, 0.24, -0.01, -0.09, 0.19, 0.21, 0.14, 0.08, -0.21, -0.07, -0.13, -0.31, 0.39, 0.23, -0.09, 0.14, 0.38, -0.47, 0.16, -0.2, -0.1, -0.04, -0.1, 0.09, 0.38, 0.19, -0.04, -0.06, 0.03, -0.33, -0.37, -0.09, 0.03, -0.25, 0.01, 0.13, -0.04, 0.18, 0.12, -0.39, 0.04, 0.1, 0.29, 0.07, 0.17, -0.24, -0.02 }, {-0.26, -0.27, -0.16, -0.58, -0.32, 0.28, -0.02, -0.56, 0.03, 0.1, -0.12, 0.08, 0.07, -0.43, 0.09, 0.7, -0.26, -0.16, -0.12, -0.41, -0.05, -0.52, -0.11, -0.27, 0.13, -0.02, 0.11, 0.01, 0.09, 0.34, -0.32, 0.18, 0.2, 0.35, 0.18, -0.2, -0.05, 0.36, -0.15, 0.11, 0.08, -0.08, 0.2, -0.38, -0.22, -0.13, 0.0, 0.04, -0.54, 0.05, 0.25, 0.14, -0.18, 0.15, 0.08, -0.35, 0.7, -0.62, -0.06, -0.07 }, {-0.2, -0.36, -0.22, -0.69, -0.18, 0.12, 0.1, -0.67, 0.03, 0.04, 0.09, 0.09, 0.05, -0.33, 0.32, 0.58, -0.27, -0.26, -0.19, -0.85, -0.29, -0.35, 0.21, -0.28, 0.01, -0.15, 0.03, 0.02, 0.06, -0.16, -0.45, 0.18, 0.26, 0.24, 0.04, -0.39, -0.05, 0.62, 0.07, -0.02, -0.02, -0.13, 0.04, -0.48, -0.19, -0.29, 0.18, -0.15, -0.48, 0.13, 0.19, -0.04, -0.15, 0.15, 0.02, -0.31, 0.35, -0.39, 0.05, -0.11 }, {0.64, 0.26, -0.39, -0.04, -0.08, -0.36, 0.04, -0.01, 0.09, 0.11, 0.25, 0.01, 0.05, 0.04, 0.11, 0.23, 0.09, -0.14, 0.02, -0.01, -0.07, -0.35, -0.35, 0.05, -0.04, 0.26, -0.23, -0.21, -0.02, 0.14, -0.32, 0.47, 0.34, -0.2, 0.06, -0.22, -0.01, 0.03, 0.24, -0.09, -0.21, 0.34, -0.44, 0.26, 0.18, -0.49, 0.04, 0.02, 0.03, 0.08, 0.22, -0.34, 0.2, -0.74, -0.03, 0.43, 0.13, 0.21, -0.22, 0.39 }, {0.6, 0.26, -0.39, -0.04, -0.03, -0.35, 0.0, -0.11, 0.1, 0.13, 0.21, 0.05, 0.15, 0.14, 0.1, 0.22, -0.01, -0.03, 0.11, 0.0, -0.15, -0.33, -0.34, 0.03, -0.02, 0.26, -0.19,

-0.21, -0.01, 0.19, -0.22, 0.44, 0.36, -0.24, 0.06, -0.25, -0.07, 0.15, 0.22, -0.15, -0.11, 0.4, -0.44, 0.36, 0.08, -0.51, 0.01, -0.02, 0.03, 0.1, 0.24, -0.24, 0.24, -0.66, -0.03, 0.42, 0.06, 0.21, -0.21, 0.27 }, {0.3, 0.36, -0.25, -0.25, -0.3, -0.3, 0.11, -0.35, 0.12, 0.12, 0.33, 0.12, 0.1, 0.18, -0.17, -0.06, 0.03, 0.61, -0.02, -0.31, 0.15, -0.06, -0.17, -0.15, 0.22, -0.13, -0.08, -0.43, 0.03, 0.15, 0.09, 0.51, -0.13, -0.43, -0.21, -0.17, 0.12, 0.08, -0.47, 0.1, -0.2, 0.28, -0.4, 0.51, 0.25, -0.01, 0.03, -0.16, -0.3, 0.12, 0.29, -0.1, -0.14, 0.25, -0.06, 0.05, 0.27, -0.21, -0.24, 0.26 }, {0.5, -0.22, -0.08, -0.23, -0.35, 0.01, 0.11, -0.23, 0.18, 0.03, -0.04, 0.04, 0.02, 0.04, -0.32, 0.26, -0.22, 0.09, -0.33, -0.27, 0.09, -0.2, 0.13, -0.08, -0.31, -0.21, 0.55, 0.0, -0.4, 0.17, 0.18, 0.8, 0.29, -0.47, -0.02, -0.07, 0.08, 0.03, -0.25, -0.21, -0.24, 0.27, -0.39, 0.13, 0.08, 0.11, 0.34, -0.35, -0.31, -0.0, 0.19, -0.38, -0.24, 0.12, -0.09, 0.63, 0.37, -0.1, -0.18, 0.34 }, {0.26, -0.49, 0.1, 0.39, -0.53, 0.43, 0.39, 0.06, 0.17, 0.01, -0.46, 0.08, 0.05, 0.13, -0.54, -0.01, -0.07, 0.3, -0.6, 0.62, -0.16, -0.28, 0.42, -0.2, -0.33, -0.57, 0.39, -0.49, -0.49, 0.34, -0.16, -0.02, 0.14, -0.37, 0.29, -0.29, -0.0, -0.4, -0.4, -0.11, 0.46, 0.18, 0.09, -0.06, -0.18, 0.32, 0.92, -0.61, -0.67, 0.09, 0.01, -0.9, -0.19, 0.04, -0.16, 0.18, -0.21, -0.25, 0.04, 0.0 }, {0.19, -0.31, -0.05, 0.56, -0.36, 0.03, 0.32, 0.23, -0.04, -0.05, -0.11, -0.01, -0.06, 0.22, -0.48, 0.34, 0.17, 0.43, -0.57, 0.53, -0.11, -0.67, 0.38, -0.22, -0.07, -0.76, 0.44, -0.62, -0.21, 0.22, 0.02, -0.01, 0.02, -0.23, 0.4, -0.29, -0.05, -0.76, -0.51, -0.24, 0.22, 0.19, 0.12, -0.04, 0.03, 0.2, 0.77, -0.57, -0.81, -0.05, -0.22, -0.77, -0.21, -0.11, -0.33, 0.27, 0.01, -0.11, 0.01, -0.03 }, {0.13, -0.21, 0.03, -0.19, -0.22, 0.01, 0.17, 0.12, -0.0, -0.02, -0.05, 0.05, -0.04, -0.08, 0.08, 0.02, -0.07, 0.21, -0.43, -0.15, -0.09, -0.51, 0.63, -0.38, -0.3, 0.04, 0.12, -0.29, -0.09, 0.2, 0.22, 0.24, 0.01, -0.28, 0.13, -0.35, -0.08, -0.0, 0.13, -0.41, 0.47, -0.02, 0.12, 0.02, 0.05, -0.57, 0.48, -0.36, -0.28, -0.08, 0.35, -0.31, -0.3, -0.39, -0.22, 0.22, 0.51, 0.0, -0.01, 0.18 }, {0.54, -0.55, 0.22, -0.04, -0.4, 0.1, -0.14, 0.21, 0.03, -0.02, -0.12, 0.09, -0.01, 0.29, -0.33, -0.1, 0.29, 0.01, -0.34, -0.17, -0.29, -0.26, 0.47, -0.27, -0.28, -0.09, -0.18, -0.24, -0.32, -0.29, 0.02, -0.11, 0.28, 0.24,

0.04, -0.52, -0.11, -0.01, 0.02, -0.07, 0.12, 0.04, 0.37, -0.19, -0.18, -0.61, 0.4, -0.01, 0.23, 0.05, 0.12, -0.1, -0.01, -0.34, -0.12, 0.1, 0.39, -0.08, 0.3, -0.31 }, {0.2, -0.08, -0.1, -0.48, -0.33, -0.12, -0.19, -0.04, -0.12, 0.02, -0.11, 0.06, -0.02, -0.25, 0.18, 0.16, -0.03, -0.12, -0.28, -0.39, -0.35, -0.02, -0.15, -0.11, 0.24, 0.09, -0.01, 0.38, -0.14, -0.0, 0.33, 0.04, 0.49, 0.09, 0.16, -0.15, -0.02, 0.01, -0.08, -0.33, 0.24, 0.02, 0.22, 0.24, 0.11, -0.02, -0.24, -0.04, 0.22, 0.02, 0.2, 0.16, -0.2, -0.29, -0.01, 0.24, 0.34, -0.01, 0.19, 0.09 }, {0.31, 0.01, -0.06, -0.31, -0.32, 0.33, 0.07, 0.22, -0.05, 0.03, -0.21, -0.0, -0.02, -0.39, 0.17, 0.39, -0.12, -0.43, -0.24, 0.04, -0.54, -0.06, -0.25, 0.0, 0.17, 0.08, 0.09, 0.27, 0.02, -0.04, 0.15, 0.11, 0.45, 0.36, 0.25, -0.21, -0.04, -0.2, -0.38, -0.27, 0.31, -0.07, 0.22, -0.01, 0.1, 0.07, -0.06, 0.27, 0.17, -0.0, -0.03, 0.01, -0.17, -0.4, -0.01, 0.15, 0.31, -0.03, 0.11, -0.09 }, {-0.26, -0.34, -0.08, -0.24, -0.08, 0.53, -0.05, -0.39, -0.05, -0.06, -0.22, -0.01, 0.03, -0.4, 0.28, 0.3, -0.36, -0.13, -0.12, -0.5, -0.15, 0.06, -0.38, 0.33, 0.21, 0.33, 0.27, 0.35, -0.1, -0.49, 0.15, 0.28, 0.2, 0.64, -0.07, -0.06, 0.06, 0.25, 0.16, -0.18, -0.3, 0.06, -0.1, 0.12, 0.06, -0.14, -0.26, 0.12, 0.09, 0.07, -0.09, 0.62, -0.12, -0.19, 0.22, 0.13, 0.42, -0.19, -0.14, -0.03 }, {-0.69, -0.06, 0.03, 0.39, 0.31, -0.08, 0.16, -0.24, 0.05, 0.07, -0.14, 0.05, 0.06, 0.07, 0.09, 0.13, -0.09, 0.43, 0.11, 0.31, 0.04, 0.11, -0.34, 0.31, 0.39, 0.31, 0.52, 0.05, 0.25, -0.12, 0.39, -0.17, -0.07, 0.65, 0.19, 0.27, 0.12, 0.38, -0.23, -0.34, -0.39, -0.0, -0.21, 0.26, -0.03, 0.04, -0.38, 0.31, 0.19, 0.0, -0.7, 0.6, -0.11, -0.12, 0.19, 0.05, 0.26, -0.1, -0.06, -0.72 }, {-0.68, 0.22, 0.04, 0.38, 0.28, -0.0, 0.12, 0.18, 0.11, 0.05, 0.06, 0.08, -0.0, 0.01, -0.07, 0.14, 0.18, 0.31, 0.23, -0.1, 0.15, -0.19, -0.14, 0.1, 0.42, 0.1, 0.37, 0.08, 0.11, -0.04, -0.02, 0.03, 0.12, 0.22, 0.14, 0.35, 0.16, 0.38, -0.45, -0.17, -0.13, 0.13, -0.17, -0.16, -0.09, 0.01, -0.38, -0.07, -0.09, 0.03, -0.64, 0.38, -0.11, -0.01, 0.1, -0.17, -0.3, -0.03, -0.14, -0.39 }, {-0.2, 0.34, 0.24, 0.34, 0.1, 0.23, 0.2, 0.47, -0.04, 0.03, 0.03, 0.11, 0.05, 0.1, -0.13, 0.09, 0.36, 0.11, 0.42, 0.24, -0.11, -0.23, -0.3, 0.11, 0.09, 0.06, 0.14, -0.01, 0.13, 0.35, -0.67, -0.17, -0.23, 0.47, 0.16, -0.16, 0.06, 0.16, -0.54, -0.21, -0.0, 0.1, -0.19, -

0.68, -0.08, -0.07, -0.27, 0.25, 0.46, 0.01, -0.65, 0.33, -0.17, -0.15, 0.21, 0.24, -0.59, 0.27, -0.24, -0.58 }, {-0.2, -0.09, 0.14, -0.24, -0.0, 0.27, -0.33, 0.55, -0.11, -0.02, 0.05, 0.02, 0.07, 0.15, -0.33, 0.1, 0.53, -0.27, -0.04, 0.26, -0.07, -0.39, -0.32, -0.61, 0.45, -0.33, -0.11, -0.15, 0.06, 0.59, -0.3, -0.12, -0.35, -0.08, 0.09, -0.2, 0.06, -0.03, -0.26, 0.42, 0.44, -0.11, 0.26, -0.46, -0.15, -0.29, -0.11, 0.14, 0.07, -0.04, -0.42, -0.08, -0.09, 0.16, 0.29, -0.31, 0.12, -0.49, -0.1, -0.54 }, {-0.17, -0.19, -0.1, -0.71, 0.14, 0.03, -0.48, 0.49, -0.06, 0.02, 0.06, 0.0, -0.05, -0.21, -0.2, -0.12, 0.17, -0.57, -0.11, -0.27, -0.41, -0.16, -0.06, -0.41, 0.67, -0.36, -0.04, 0.14, 0.39, 0.21, -0.55, 0.0, 0.02, -0.31, 0.01, -0.13, 0.04, 0.51, 0.06, 0.64, 0.44, -0.19, -0.03, -0.5, -0.2, -0.67, 0.21, 0.05, 0.14, 0.03, -0.01, -0.04, 0.12, 0.4, 0.36, -0.84, 0.02, 0.02, -0.15, 0.1 }, {0.25, -0.03, -0.32, -0.31, -0.36, 0.17, -0.15, -0.57, 0.06, 0.05, 0.18, 0.0, 0.03, 0.13, -0.41, -0.08, -0.39, -0.21, -0.02, 0.01, 0.1, -0.16, -0.41, 0.01, 0.2, 0.62, -0.4, -0.28, 0.08, -0.05, -0.0, 0.23, 0.24, 0.12, -0.18, -0.51, 0.07, 0.74, 0.59, 0.33, -0.21, 0.25, 0.13, 0.33, -0.17, -0.63, -0.33, 0.26, 0.03, -0.01, -0.31, 0.18, 0.01, -0.16, -0.01, 0.08, 0.67, 0.31, 0.06, 0.05 }, {0.23, 0.1, -0.39, -0.37, -0.41, 0.08, -0.14, -0.65, 0.07, 0.08, 0.1, 0.04, 0.04, 0.16, -0.38, -0.16, -0.4, -0.18, 0.11, -0.01, -0.0, -0.09, -0.32, -0.01, 0.05, 0.55, -0.36, -0.28, -0.01, -0.06, -0.05, 0.14, 0.24, 0.13, -0.13, -0.6, 0.09, 0.78, 0.58, 0.43, -0.13, 0.34, 0.12, 0.36, -0.21, -0.55, -0.35, 0.24, 0.05, -0.0, -0.37, 0.11, 0.09, -0.09, -0.04, 0.01, 0.71, 0.3, 0.04, 0.08 }, {0.28, -0.21, -0.17, -0.06, -0.24, 0.01, 0.2, -0.32, 0.06, -0.0, 0.12, -0.0, 0.01, 0.1, -0.23, -0.23, -0.01, 0.3, -0.28, 0.18, 0.13, -0.12, 0.35, -0.15, 0.02, -0.09, 0.12, -0.44, 0.08, -0.5, 0.08, 0.15, -0.09, 0.15, -0.08, -0.19, 0.06, 0.4, -0.04, 0.1, 0.2, 0.24, 0.06, 0.33, -0.0, -0.07, 0.32, -0.27, -0.46, -0.01, -0.04, -0.27, -0.47, 0.05, 0.12, 0.14, 0.1, 0.01, 0.12, -0.06 }, {0.07, -0.68, -0.06, -0.63, -0.56, 0.28, 0.02, -0.3, 0.15, 0.04, -0.08, 0.11, 0.07, -0.23, -0.39, 0.28, -0.56, -0.09, -0.77, 0.3, 0.24, -0.15, 0.32, 0.01, 0.09, -0.16, 0.19, 0.13, 0.01, -0.04, 0.19, 0.48, 0.25, -0.23, -0.07, -0.19, -0.0, 0.66, 0.18, 0.07, 0.53, 0.28, -0.17, 0.26, -0.13, 0.19, 0.62, -0.0, -0.43, -0.01,

0.3, -0.61, -0.37, 0.21, 0.3, -0.01, 0.63, 0.28, -0.14, 0.41 }, {0.18, -0.34, -0.03, -0.31, -0.47, -0.04, 0.21, 0.11, 0.09, -0.02, -0.27, 0.1, -0.05, -0.22, -0.14, 0.36, -0.04, 0.39, -0.69, 0.15, 0.17, -0.49, 0.4, -0.29, -0.14, -0.24, 0.23, -0.3, -0.17, 0.17, 0.13, 0.07, -0.05, -0.15, 0.26, -0.3, -0.06, -0.21, -0.05, -0.51, 0.57, 0.21, -0.33, -0.15, -0.13, 0.2, 0.47, -0.29, -0.59, -0.04, 0.38, -0.76, -0.08, -0.15, -0.23, 0.48, -0.15, -0.12, -0.2, -0.02 }, {0.5, -0.19, -0.02, -0.24, -0.57, -0.09, -0.07, 0.25, -0.01, 0.0, -0.12, 0.1, 0.01, 0.18, -0.22, 0.27, 0.18, 0.69, -0.4, 0.42, 0.37, -0.47, 0.56, -0.43, -0.53, -0.34, 0.25, -0.39, -0.29, 0.45, 0.08, 0.15, -0.13, -0.57, 0.12, -0.67, -0.08, -0.39, -0.21, -0.24, 0.7, 0.21, -0.19, -0.2, -0.03, 0.2, 0.23, -0.48, -0.45, -0.03, 0.36, -0.97, -0.27, 0.04, -0.14, 0.4, -0.2, -0.21, -0.41, 0.37 }, {0.05, -0.24, -0.02, -0.45, 0.06, -0.25, -0.27, -0.17, -0.05, -0.05, -0.06, 0.09, 0.03, -0.61, -0.05, 0.02, -0.19, 0.54, -0.44, 0.04, -0.19, -0.39, 0.49, -0.57, -0.03, 0.06, 0.28, -0.13, -0.05, 0.28, 0.33, 0.54, -0.15, -0.42, -0.01, -0.24, -0.11, -0.15, 0.5, -0.0, 0.09, 0.19, -0.17, 0.12, -0.01, -0.4, 0.58, -0.31, -0.2, -0.01, 0.63, -0.37, -0.41, -0.19, -0.03, 0.22, 0.51, -0.17, -0.13, 0.48 }, {0.45, -0.43, 0.2, -0.49, -0.11, -0.05, -0.37, 0.12, -0.07, 0.01, -0.11, 0.03, -0.0, -0.54, -0.19, -0.23, -0.1, 0.24, -0.43, 0.13, -0.04, -0.07, 0.36, -0.38, -0.2, 0.01, 0.07, -0.09, -0.23, 0.23, 0.03, 0.31, -0.09, -0.4, -0.05, -0.4, -0.09, -0.21, 0.39, 0.02, 0.13, 0.09, -0.03, -0.02, -0.19, -0.29, 0.59, -0.21, 0.07, -0.1, 0.74, -0.19, -0.07, 0.13, 0.3, -0.06, 0.45, -0.05, -0.15, 0.57 }, {0.23, -0.05, 0.24, -0.38, -0.34, 0.04, -0.01, -0.13, -0.12, -0.01, -0.09, 0.11, -0.06, -0.55, 0.38, -0.02, -0.23, 0.01, -0.25, 0.07, -0.44, 0.01, 0.23, -0.24, 0.25, -0.02, 0.29, 0.39, -0.19, 0.3, 0.18, 0.39, 0.18, 0.2, -0.09, -0.22, -0.12, 0.21, 0.36, -0.13, 0.07, 0.08, 0.36, 0.2, 0.02, 0.28, 0.2, -0.28, -0.2, -0.01, 0.26, 0.02, -0.28, 0.13, -0.09, 0.05, 0.48, 0.17, 0.36, 0.17 }, {0.15, -0.0, 0.13, -0.23, -0.4, 0.11, 0.15, 0.15, -0.14, -0.03, -0.15, 0.06, 0.0, -0.56, 0.27, 0.23, -0.28, -0.12, -0.05, 0.35, -0.47, -0.06, -0.11, -0.27, 0.38, 0.05, 0.16, 0.26, 0.0, 0.29, 0.13, 0.37, 0.05, 0.34, 0.07, -0.24, -0.07, 0.13, 0.03, -0.04, -0.03, 0.04, 0.37, -0.05, 0.05, 0.2, 0.26, 0.04, -0.15, 0.0, 0.13, -0.08, -0.27, -0.03, -0.17, 0.05, 0.6,

0.32, 0.18, -0.03 }, {-0.33, 0.03, -0.23, -0.12, 0.01, 0.46, -0.2, -0.24, -0.04, -0.07, -0.14, 0.11, 0.01, -0.37, 0.14, 0.06, -0.06, -0.12, -0.12, -0.12, -0.27, 0.08, -0.48, -0.04, 0.58, 0.3, 0.26, 0.3, -0.11, 0.01, 0.14, 0.21, 0.1, 0.3, 0.11, 0.1, 0.02, 0.17, 0.15, 0.14, -0.12, 0.15, -0.02, 0.09, 0.01, 0.22, 0.02, -0.0, -0.18, -0.01, 0.08, 0.33, -0.25, 0.04, 0.06, -0.15, 0.57, 0.25, -0.0, 0.06 }, {0.06, 0.27, -0.16, -0.02, -0.31, 0.34, -0.04, -0.05, 0.07, -0.01, -0.09, -0.0, -0.06, 0.01, -0.25, -0.22, -0.02, 0.13, 0.07, 0.07, -0.18, 0.15, -0.54, 0.16, 0.0, 0.01, -0.01, 0.12, 0.15, 0.48, -0.29, -0.23, -0.18, 0.36, 0.24, -0.12, -0.02, 0.08, -0.31, 0.26, -0.29, 0.13, -0.0, -0.36, -0.17, 0.3, -0.26, 0.7, 0.02, -0.06, -0.6, 0.53, -0.14, 0.17, 0.21, -0.35, -0.04, 0.06, -0.04, -0.46 }, {-0.21, 0.42, -0.04, 0.25, 0.01, 0.58, 0.16, 0.2, -0.05, 0.07, -0.15, -0.01, 0.03, -0.18, -0.22, 0.11, -0.17, 0.09, 0.19, -0.19, 0.1, 0.06, -0.16, 0.29, 0.09, 0.35, -0.11, -0.07, 0.12, 0.17, -0.24, 0.02, -0.02, 0.62, 0.1, -0.22, 0.1, 0.04, -0.27, 0.18, -0.09, 0.21, 0.44, -0.0, -0.3, 0.11, -0.16, 0.48, 0.24, 0.0, -0.57, 0.51, 0.15, 0.23, 0.25, -0.2, -0.04, -0.09, -0.04, -0.61 }, {0.19, 0.14, -0.01, -0.06, -0.42, 1.06, 0.14, 0.34, -0.01, 0.1, -0.07, 0.02, 0.05, -0.58, -0.32, 0.08, -0.25, -0.12, 0.05, 0.16, -0.4, 0.2, -0.39, 0.29, -0.0, -0.31, -0.48, -0.0, 0.25, 0.28, -0.34, 0.16, -0.04, 0.39, 0.13, -0.41, 0.07, -0.46, -0.57, 0.5, 0.33, 0.16, 0.27, -0.38, -0.28, 0.09, 0.29, 0.7, 0.36, 0.05, -0.68, 0.21, 0.24, 0.52, 0.35, -0.53, -0.19, 0.18, -0.04, -0.36 }, {-0.22, -0.01, 0.17, -0.42, 0.02, -0.01, -0.44, 0.66, 0.02, 0.06, -0.1, 0.05, 0.09, -0.08, -0.31, -0.03, -0.29, -0.45, 0.22, 0.16, -0.57, -0.31, -0.56, -0.31, 0.84, -0.02, -0.53, -0.29, 0.99, 0.4, 0.23, -0.26, -0.39, -0.49, 0.19, -0.14, 0.08, 0.09, 0.25, 0.93, 0.82, -0.14, 0.38, 0.2, -0.32, -0.33, 0.35, 0.79, 0.66, 0.1, -0.78, -0.33, 0.24, 0.09, 0.13, -0.73, 0.08, 0.54, 0.39, -0.42 }, {-0.21, -0.25, 0.27, -0.49, 0.11, -0.14, -0.8, 0.62, 0.04, 0.08, -0.25, 0.06, 0.05, -0.31, -0.6, -0.24, -0.34, -0.74, 0.2, -0.45, -0.74, -0.1, -0.07, -0.36, 1.09, 0.18, -0.6, -0.15, 0.95, -0.04, 0.28, -0.48, -0.06, -0.43, -0.15, 0.04, 0.06, 0.58, 0.8, 1.13, 0.79, -0.02, 0.35, 0.35, -0.3, -0.58, 0.28, 0.74, 0.75, 0.03, -0.73, -0.08, 0.42, 0.42, 0.16, -0.82, 0.03, 0.52, 0.51, -0.33 }, {0.4, -0.28, -0.18, -0.47, -0.35, -0.1,

0.02, -0.83, 0.18, 0.05, 0.3, 0.13, 0.06, 0.42, -0.22, 0.8, -0.16, -0.04, 0.1, -0.19, -0.34, -0.24, -0.7, -0.09, 0.8, 0.29, -0.81, -0.63, 0.25, 0.04, 0.14, 0.31, 0.28, 0.39, 0.13, -0.26, 0.1, 0.67, 0.37, 0.2, 0.25, -0.11, 0.43, 0.56, 0.03, -0.94, -0.47, 0.47, 0.03, -0.03, -1.1, -0.26, 0.2, 0.16, -0.73, -0.92, 0.96, 0.23, 0.34, 0.25 }, {0.5, -0.15, -0.12, -0.4, -0.35, -0.17, 0.04, -0.7, 0.22, 0.04, 0.33, 0.12, 0.13, 0.41, -0.18, 0.77, -0.04, -0.04, 0.11, -0.15, -0.3, -0.19, -0.61, -0.13, 0.61, 0.37, -0.84, -0.64, -0.1, 0.02, 0.07, 0.26, 0.26, 0.37, 0.07, -0.38, 0.06, 0.63, 0.34, 0.07, 0.22, -0.05, 0.34, 0.5, 0.05, -0.84, -0.53, 0.34, 0.11, 0.01, -1.18, -0.25, 0.18, -0.14, -0.7, -0.54, 0.82, 0.22, 0.32, 0.18 }, {0.85, -0.24, -0.37, -0.36, -0.8, -0.15, 0.3, -0.33, 0.13, 0.03, 0.55, 0.05, 0.03, 0.19, -0.25, 1.15, -0.11, 0.24, 0.18, -0.48, -0.31, -0.28, -0.59, -0.28, 0.54, 0.12, -0.46, -0.77, -0.07, -0.35, -0.04, 0.63, 0.12, 0.7, 0.04, -0.75, 0.01, 0.83, 0.38, -0.14, 0.33, -0.03, 0.35, 0.24, -0.09, -0.7, -0.22, 0.4, -0.19, -0.07, -1.47, -0.26, -0.13, -0.23, -0.78, -0.09, 0.55, -0.15, 0.32, 0.26 }, {0.67, -0.35, -0.15, -0.31, 0.02, -0.22, -0.18, 0.12, 0.06, -0.1, 0.13, 0.07, -0.0, 0.55, -0.09, 0.1, -0.23, -0.5, 0.11, -0.0, 0.26, 0.13, 0.25, -0.12, 0.2, 0.69, -0.34, 0.02, -0.3, -0.23, 0.1, 0.6, 0.25, -0.63, -0.07, -0.64, -0.12, 0.67, 0.84, -0.35, 0.06, 0.15, -0.57, 0.11, -0.2, -0.33, -0.26, 0.07, 0.23, -0.06, -0.26, -0.34, -0.16, -0.59, 0.0, 0.55, 0.58, 0.36, -0.26, 0.26 }, {0.5, 0.24, -0.3, -0.57, -0.22, 0.02, 0.22, -0.04, 0.05, 0.05, -0.13, 0.03, 0.02, 0.14, -0.15, 0.41, -0.13, 0.21, -0.19, 0.05, -0.22, -0.54, 0.04, -0.29, -0.08, 0.31, -0.1, -0.31, -0.23, 0.36, 0.25, 0.29, 0.19, 0.01, 0.39, -0.8, -0.18, 0.22, 0.23, -0.47, 0.57, 0.25, -0.32, 0.11, -0.34, 0.11, 0.04, 0.02, -0.14, -0.01, 0.02, -0.45, -0.22, -0.35, -0.19, 0.67, 0.68, -0.13, -0.07, -0.18 }, {0.47, -0.27, -0.17, -0.64, -0.57, 0.03, 0.23, -0.04, 0.12, -0.05, -0.28, 0.07, 0.04, -0.01, -0.08, 0.54, 0.07, 0.45, -0.45, 0.26, 0.05, -0.46, 0.3, -0.27, -0.3, 0.04, 0.18, -0.15, -0.39, 0.25, 0.39, 0.05, 0.22, -0.34, 0.3, -0.76, 0.02, 0.2, -0.1, -0.43, 0.57, 0.24, -0.17, -0.09, -0.32, 0.22, 0.24, -0.12, -0.47, -0.02, 0.22, -0.63, -0.51, -0.2, -0.09, 0.39, 0.56, -0.08, -0.23, 0.39 }, {0.27, -0.31, -0.1, -0.82, -0.41, -0.2, 0.12, -0.29, 0.08, 0.05, -0.27, 0.06, 0.03, -0.64,

-0.05, 0.63, -0.13, 0.49, -0.89, 0.02, -0.1, -0.23, 0.37, -0.37, -0.13, 0.28, 0.46, 0.2, -0.21, 0.38, 0.29, 0.48, 0.23, 0.08, 0.23, -0.44, -0.08, 0.34, 0.49, -0.34, 0.08, 0.13, 0.07, 0.04, -0.21, 0.1, 0.8, -0.28, -0.61, 0.07, 0.68, -0.36, -0.48, -0.26, 0.1, 0.53, 0.7, -0.2, 0.06, 0.4 }, {0.57, -0.54, -0.08, -0.87, -0.62, -0.06, -0.16, -0.01, -0.06, -0.01, -0.37, 0.04, 0.1, -0.65, -0.11, 0.4, -0.12, 0.18, -0.88, -0.16, -0.09, -0.29, 0.28, -0.31, -0.36, 0.36, 0.26, 0.34, -0.21, 0.49, -0.0, 0.27, 0.09, 0.03, -0.02, -0.47, -0.08, 0.22, 0.67, -0.07, 0.31, 0.18, -0.08, -0.03, -0.36, 0.14, 0.83, -0.16, -0.26, -0.01, 0.83, -0.04, -0.25, -0.3, 0.41, 0.61, 0.32, -0.04, -0.25, 0.64 }, {0.72, -0.06, 0.13, -0.6, -0.71, 0.06, 0.05, -0.11, -0.17, 0.01, -0.28, 0.13, 0.04, -0.88, 0.3, 0.25, -0.2, 0.31, -0.75, 0.06, -0.27, -0.18, 0.19, -0.13, -0.49, 0.48, 0.38, 0.49, -0.51, 0.67, -0.3, 0.24, 0.03, -0.06, -0.01, -0.44, 0.06, 0.34, 0.16, -0.24, 0.61, 0.26, 0.2, -0.04, -0.2, 0.56, 0.68, -0.19, -0.54, -0.02, 0.69, -0.1, -0.43, -0.43, -0.11, 0.41, 0.25, -0.08, 0.14, 0.37 }, {0.55, 0.21, -0.01, -0.66, -0.67, 0.02, 0.11, 0.25, -0.08, -0.04, -0.25, 0.11, 0.09, -0.83, 0.24, 0.27, -0.08, 0.12, -0.64, 0.52, -0.16, -0.2, -0.1, -0.19, -0.15, 0.4, 0.44, 0.33, -0.27, 0.64, -0.34, 0.17, -0.03, -0.04, 0.12, -0.45, -0.04, 0.13, -0.08, -0.21, 0.52, 0.12, 0.12, -0.17, -0.17, 0.4, 0.63, -0.01, -0.55, -0.01, 0.73, -0.17, -0.52, -0.3, -0.13, 0.27, 0.49, -0.02, -0.1, 0.44 }, {0.22, 0.25, -0.0, -0.75, -0.19, 0.33, -0.18, -0.03, -0.15, 0.01, -0.15, 0.12, 0.08, -0.83, 0.33, 0.19, -0.02, 0.27, -0.37, 0.48, -0.13, -0.04, -0.19, -0.15, 0.29, 0.46, 0.38, 0.41, -0.18, 0.87, 0.12, 0.23, 0.06, -0.03, 0.26, -0.27, -0.04, 0.24, -0.1, -0.06, 0.38, 0.16, 0.09, -0.16, -0.15, 0.39, 0.23, -0.22, -0.38, -0.02, 0.42, -0.42, -0.34, -0.28, 0.04, -0.07, 0.21, 0.29, -0.2, 0.75 }, {0.06, 0.03, 0.0, -0.58, -0.21, 0.35, -0.09, 0.11, 0.02, -0.01, -0.37, 0.1, 0.02, -0.66, -0.0, -0.08, -0.0, -0.02, -0.33, 0.44, -0.18, -0.14, -0.13, -0.02, 0.3, 0.18, -0.07, 0.34, 0.21, 1.13, 0.25, -0.15, -0.37, 0.25, 0.57, -0.19, 0.03, 0.35, -0.2, 0.35, 0.25, -0.01, -0.01, -0.25, -0.34, 0.19, 0.07, 0.66, -0.08, 0.04, -0.49, 0.13, -0.12, -0.24, 0.23, -0.12, 0.01, 0.25, 0.11, -0.18 }, {-0.48, 0.67, -0.09, -0.21, 0.72, 0.09, -0.18, 0.48, -0.1, 0.11, 0.06, 0.02, -0.07, -0.33, -0.29, -0.8, -0.16, -0.58, 0.42, 0.09, -

0.02, 0.01, -0.37, 0.27, 0.25, 0.57, -0.77, -0.14, 0.32, 0.39, 0.02, -0.24, -0.33, -0.17, 0.22, -0.47, -0.01, -0.12, 0.45, 0.83, 0.25, 0.16, -0.04, 0.02, -0.3, -0.24, -0.32, 0.65, 0.73, 0.04, -0.59, 0.25, 0.38, -0.11, 0.26, -0.05, -0.29, 0.41, -0.08, -0.59 }, {-0.23, 0.48, -0.12, -0.19, 0.54, 0.66, -0.12, 0.25, 0.09, 0.04, 0.15, 0.08, 0.03, -0.08, -0.3, -0.4, -0.55, -0.47, 0.27, 0.03, 0.29, 0.32, -0.68, 0.59, 0.91, 0.19, -0.62, -0.29, 0.78, 0.03, -0.04, -0.07, 0.45, -0.11, 0.15, -0.21, 0.12, 0.17, 0.36, 1.12, -0.1, 0.14, -0.02, -0.15, -0.45, -0.17, -0.41, 1.03, 0.73, 0.01, -0.46, 0.54, 0.35, 0.5, 0.5, -0.71, 0.42, 0.5, -0.04, -0.54 }, {-0.07, 0.31, 0.28, -0.29, 0.09, -0.14, -0.22, 0.4, 0.12, 0.07, 0.32, 0.02, 0.15, 0.22, -0.67, 0.48, -0.43, -0.15, 0.96, -0.86, -0.45, -0.46, -0.94, -0.26, 1.32, 0.26, -0.51, -0.47, 1.32, -0.08, -0.36, -0.4, 0.49, 0.14, 0.34, -0.51, 0.15, 0.52, 0.31, 1.47, 0.37, -0.04, 0.23, -0.18, -0.51, -0.62, -0.48, 1.02, 1.05, 0.12, -1.11, 0.14, 0.08, -0.3, -0.2, -0.45, 0.5, 1.01, 0.47, -0.88 }, {-0.26, -0.25, 0.61, -0.35, -0.16, 0.16, -0.36, -0.33, 0.15, 0.13, 0.09, 0.11, 0.07, -0.25, -0.69, 0.39, -0.63, -0.21, 0.62, -1.33, -0.66, -0.45, -0.59, -0.38, 1.44, 0.08, -0.56, -0.21, 1.17, -0.57, -0.18, -0.54, 0.77, 0.47, 0.15, -0.32, 0.3, 0.72, 0.62, 1.48, 0.73, -0.0, 0.45, -0.03, -0.45, -0.49, -0.21, 0.9, 0.69, 0.11, -1.28, -0.15, 0.33, -0.23, -0.38, -0.35, 0.3, 0.8, 0.66, -0.92 } };

Bobot pada *output layer* { {-0.07, -0.58, 0.67, -2.03, -0.78, 0.78, 0.36, 0.62, 0.74, -0.61 }, {-1.33, -1.52, 0.61, 0.41, 1.17, -1.24, -1.55, 1.02, 1.05, -1.26 }, {-0.03, -0.14, -0.59, -0.69, -0.7, -0.46, -0.62, -0.35, 0.46, 0.31 }, {1.59, -0.68, -0.97, -1.41, 0.77, -0.72, -1.51, 0.91, -0.35, 0.97 }, {-1.08, -0.18, -1.32, 1.2, 0.37, -1.15, -1.3, -1.0, 1.81, 0.39 }, {0.66, 0.31, -0.78, -0.61, 0.6, 1.16, 0.81, -1.03, -1.11, -1.28 }, {0.57, -1.26, -0.57, -0.4, 0.52, 1.81, -1.77, 0.75, -1.57, -2.14 }, {0.45, -1.75, -1.0, -1.32, 1.14, -1.99, 0.85, -0.98, 0.85, -2.2 }, {-0.13, -0.24, -0.41, -0.48, -0.45, -0.07, 0.02, -0.78, -0.36, 0.07 }, {-0.17, -0.37, -0.4, -0.41, -0.3, -0.26, -0.26, -0.22, -0.33, -0.16 }, {-0.98, -1.89, 1.61, -1.18, -0.42, -0.27, -0.46, 1.66, -1.06, -0.38 }, {-0.39, -0.34, -0.3, -0.37, -0.27, -0.22, -0.18, -0.31, -0.61, 0.03 }, {-0.27, -0.3, -0.48, -0.45, -0.38, -0.01, -0.21, -0.58, -0.5, -0.17

}, {0.92, -1.19, 0.42, -1.59, 0.57, -1.18, -0.98, 0.4, 1.22, 1.16 }, {-0.78, -0.97, -0.93, 1.19, -0.35, 1.7, -2.24, 0.55, 1.16, -1.49 }, {0.92, -1.16, 0.74, -0.42, -0.85, 0.28, -1.11, -0.9, -1.38, 0.43 }, {-0.23, -0.65, -0.96, -1.39, 0.46, -0.97, 0.24, -0.86, 0.12, 0.41 }, {-0.28, -1.29, -0.16, -0.19, -1.26, -0.36, -0.3, 0.21, -0.73, 0.2 }, {-0.52, -0.8, 0.91, -1.41, 1.24, -0.69, -1.93, 0.63, 1.46, -0.29 }, {1.09, -1.76, -1.28, 0.86, 0.8, -1.44, 0.94, -1.06, -1.26, -1.63 }, {-1.45, -1.57, -0.8, 0.27, 0.1, -0.36, 1.08, -0.93, 0.49, 0.54 }, {-1.11, 0.3, -1.01, -0.69, 0.04, 0.27, 0.66, 0.03, 0.12, -0.2 }, {0.1, 0.12, -1.07, 0.18, -1.16, -0.46, 0.29, 0.19, 0.51, 0.23 }, {-0.7, 0.55, -0.75, -0.64, 1.22, 1.35, -0.81, 0.57, -0.57, -0.81 }, {-0.67, 0.11, 0.3, 0.32, 0.68, -0.92, -0.09, -1.12, -1.53, 0.11 }, {-0.53, 0.48, -0.1, -0.83, -0.71, 0.36, -1.56, -1.08, 1.47, 0.06 }, {-0.49, -1.58, -0.93, 0.82, 0.34, 0.01, -0.62, -0.96, -0.82, 0.19 }, {-1.17, 1.06, -1.22, 0.42, -0.62, 0.32, -1.0, -0.17, 0.34, -0.89 }, {0.26, -0.07, 1.07, 1.87, 0.85, -1.5, -0.67, 0.5, -0.74, -1.19 }, {0.8, -2.09, 1.0, 0.86, -0.06, -1.55, 0.44, -1.29, -0.51, -1.85 }, {-0.43, 0.45, 0.12, 1.15, -1.81, -1.8, -0.19, 0.19, -2.36, 1.61 }, {-1.45, -1.06, 1.22, 1.12, -0.76, 1.07, -1.59, 0.9, -2.14, -1.79 }, {-0.25, -0.6, -0.38, -0.22, -0.09, 0.02, -0.32, -0.32, -0.69, -0.1 }, {0.73, 0.2, 0.3, -1.04, -0.3, 0.35, -1.45, -0.05, -0.54, 0.31 }, {1.19, -0.36, 0.42, 0.36, 0.08, 0.26, -1.41, -0.8, -0.86, -1.81 }, {-0.6, 0.4, -1.68, 1.72, 0.37, -1.41, -0.4, 0.38, -1.32, 0.4 }, {-0.09, -0.37, -0.55, -0.56, -0.41, -0.49, -0.05, -0.42, -0.23, -0.1 }, {-1.83, 0.6, 0.42, 0.26, -1.05, 0.78, -1.62, -1.83, 0.86, 1.09 }, {-1.17, 1.16, 0.06, 0.46, -0.93, -0.63, -1.03, -0.96, 1.03, 0.29 }, {-0.59, 0.82, 0.93, -0.81, 0.56, -1.54, 1.68, -1.05, -0.42, -0.44 }, {0.96, 0.88, 0.48, 0.34, -1.46, -1.42, 1.13, -0.37, 0.75, -1.33 }, {-0.61, -0.15, -0.31, -0.97, -0.55, -0.28, -0.13, -0.29, -0.76, 0.49 }, {0.23, 0.59, 0.24, -1.16, -0.94, -0.45, 0.04, 0.0, -1.1, 0.54 }, {-0.83, 0.75, -0.36, 0.32, -1.06, -0.75, 0.04, 0.09, -1.85, 0.44 }, {0.61, -0.82, -1.09, 0.08, -0.29, -1.37, 0.84, 1.46, 1.1, -1.12 }, {-0.07, -0.07, -0.56, -1.14, -0.93, 0.03, 0.09, -0.7, -1.15, -0.27 }, {0.87, 0.59, -1.14, 0.51, -1.54, 0.4, 0.66, -1.33, -1.48, -1.79 }, {-0.91, 1.77, 0.83, -1.44, 1.14, -0.06, -1.31, -1.67, -1.36, -1.61 }, {-

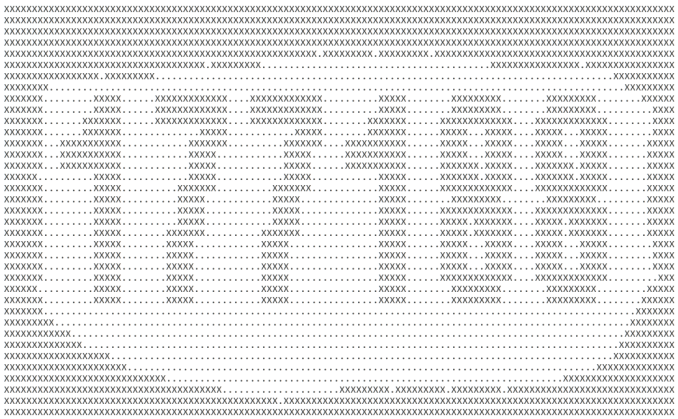
0.55, 0.93, 0.14, -1.0, 0.71, -0.87, -1.19, 0.46, 1.23, -0.7 }, {-0.24, -0.19, -0.42, -0.56, -0.29, -0.13, -0.35, -0.52, -0.5, -0.07 }, {-1.85, -0.97, -1.43, 0.89, -1.47, 0.65, 1.55, 0.55, 1.13, -0.69 }, {-1.03, 0.39, 0.11, -0.98, 0.21, 0.61, -1.18, -0.03, -0.04, 0.66 }, {0.42, 1.21, -1.54, -1.86, -1.38, -0.62, 0.89, 0.92, 0.8, -1.17 }, {-0.76, -0.92, -0.37, -0.11, 0.76, -0.15, 1.6, 0.29, -1.35, -0.32 }, {-1.3, 0.19, -0.98, -0.36, 0.47, 0.06, 0.53, -1.12, -0.25, -0.73 }, {1.04, -2.12, -0.19, -0.67, -1.18, 0.44, -2.29, -0.22, 1.49, 0.13 }, {-1.27, -1.44, 0.98, 1.16, -0.81, 0.86, -1.43, -1.05, -1.86, 0.68 }, {-0.42, 1.06, 0.31, 1.08, -0.45, -1.4, -1.28, 0.41, 1.59, -1.25 }, {1.08, 0.82, 0.12, -0.73, -1.15, -0.67, -1.03, 0.21, -1.13, 0.79 }, {-2.01, -0.84, -0.69, 0.47, -1.01, 0.82, 0.9, 0.33, 0.47, -1.5 } };

Bias pada *hidden layer* 1 {0.72, 0.99, 0.24, -0.8, 0.94, -0.79, -0.3, 0.94, -0.4, -0.31, -0.54, -0.83, -0.37, 0.98, 0.34, -1.4, 0.68, -0.19, 0.71, -0.8, 1.08, 0.81, 0.66, -0.07, -1.01, 0.69, -0.07, 0.5, -0.61, -0.71, -1.53, -0.99, -1.46, -0.59, -0.47, -0.58, -0.12, 0.2, 0.69, -0.15, -0.43, -0.49, -0.91, -1.38, 0.41, -0.04, -0.68, -0.12, 1.26, -0.49, 0.64, -0.1, 0.66, -0.65, 0.45, 0.4, -1.33, 0.91, -1.2, 0.87};

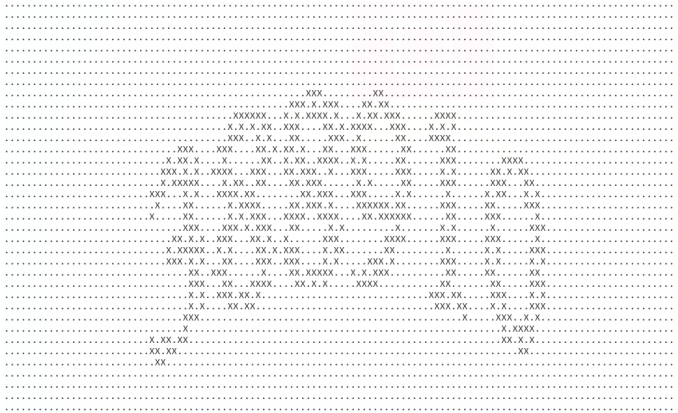
Bias pada *output layer* {-0.11, -0.3, -0.12, -0.33, -0.1, 0.01, -0.32, -0.11, -0.15, -0.11};

[*Halaman ini sengaja dikosongkan*]

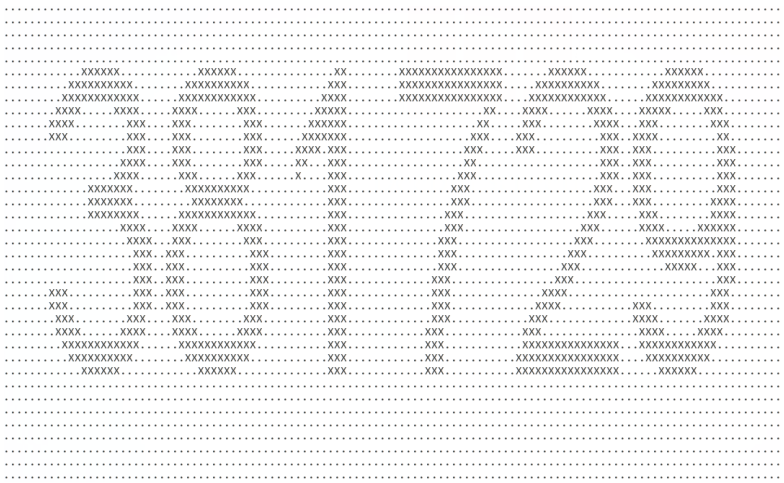
LAMPIRAN B: Data Citra Pengujian Lokal Program C++



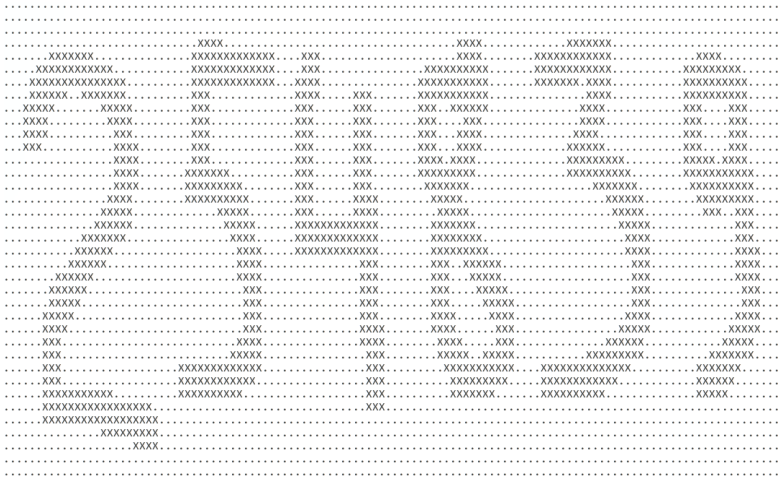
Gambar B.1 Citra 1 Pengujian Lokal Program C++



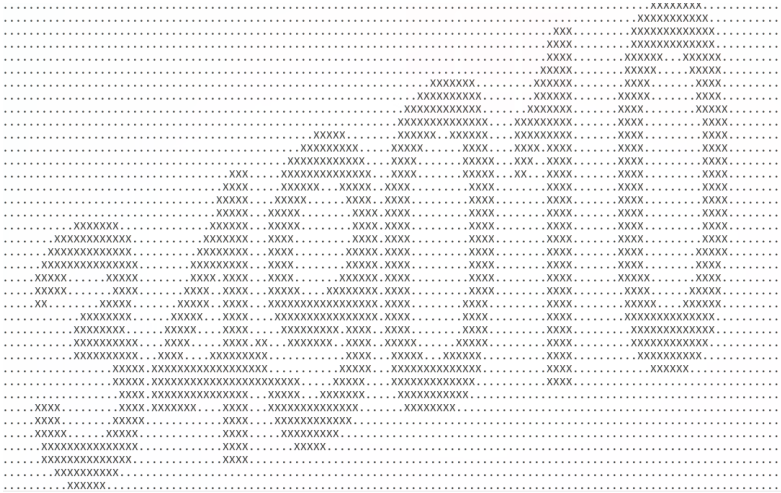
Gambar B.2 Citra 2 Pengujian Lokal Program C++



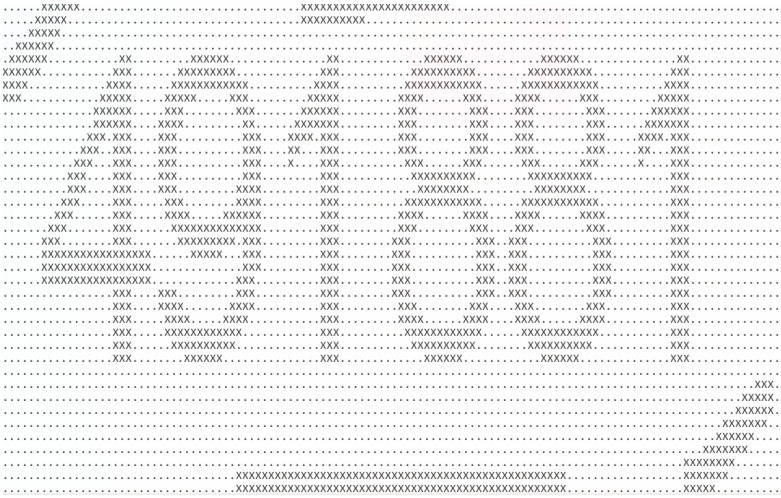
Gambar B.3 Citra 3 Pengujian Lokal Program C++



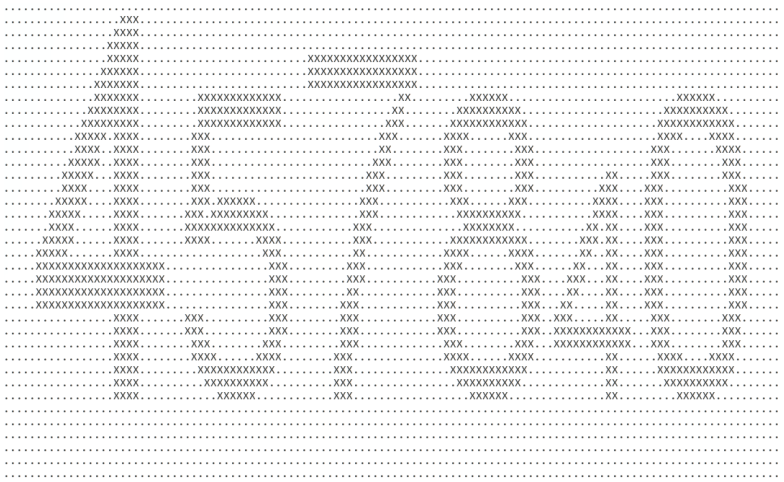
Gambar B.4 Citra 4 Pengujian Lokal Program C++



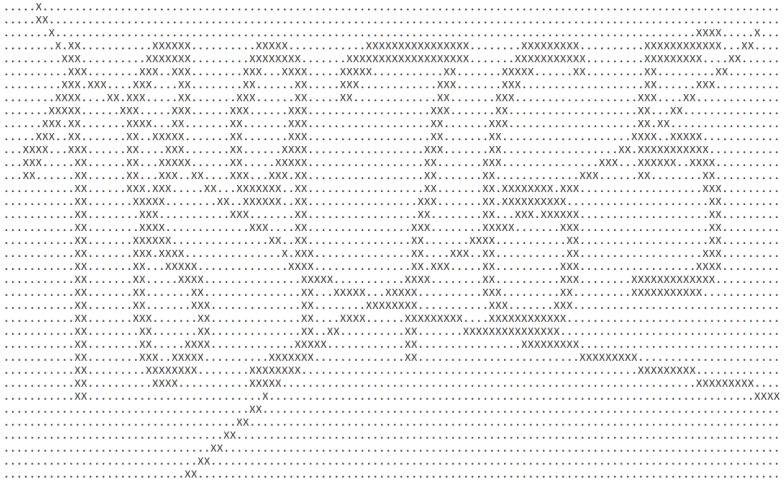
Gambar B.5 Citra 5 Pengujian Lokal Program C++



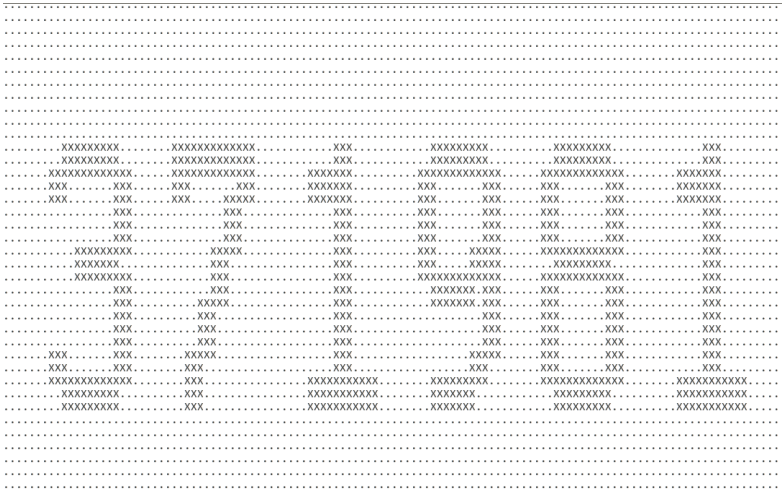
Gambar B.6 Citra 6 Pengujian Lokal Program C++



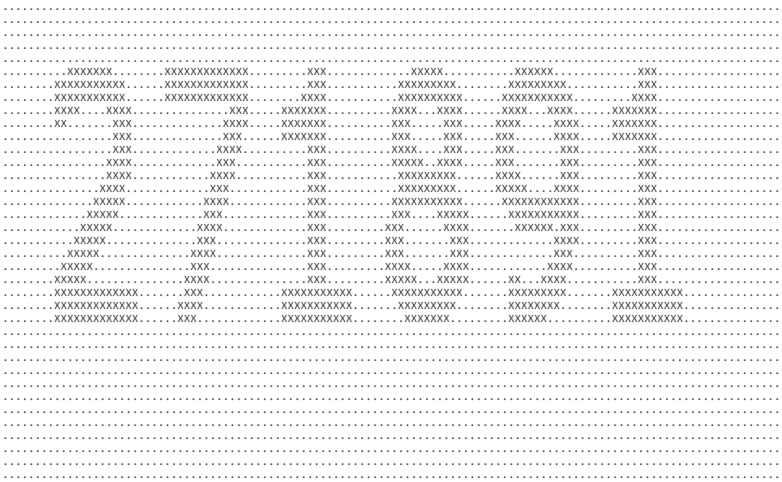
Gambar B.7 Citra 7 Pengujian Lokal Program C++



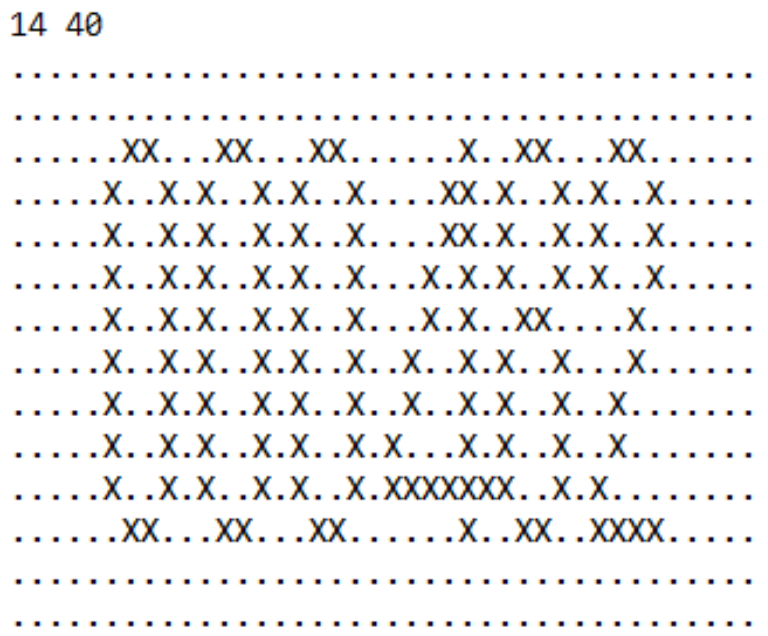
Gambar B.8 Citra 8 Pengujian Lokal Program C++



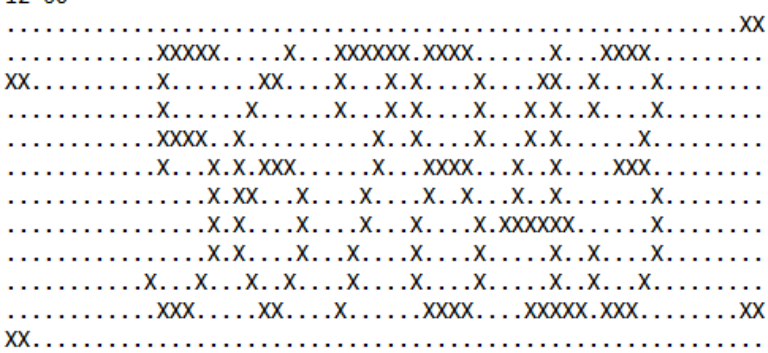
Gambar B.9 Citra 1 Pengujian Lokal Program C++



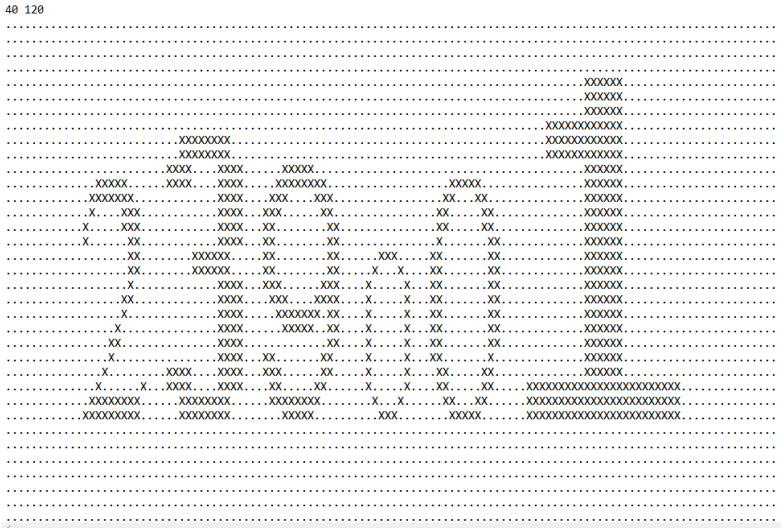
Gambar B.10 Citra 10 Pengujian Lokal Program C++



Gambar B.11 Citra 11 Pengujian Lokal Program C++



Gambar B.12 Citra 12 Pengujian Lokal Program C++



Gambar B.13 Citra 13 Pengujian Lokal Program C++

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN C: Font yang Digunakan pada Dataset

Tabel C.1 Font yang Digunakan pada Dataset (1)

No	Nama Font	Jenis Font	Ketebalan
1	Veggiemars	Scripts	Light
2	Luna	Scripts	Light
3	Babiole	Scripts	Semi-light
4	Florida	Scripts	Medium
5	Anime Ace BB	Scripts	Medium
6	Chocky	Scripts	Semi-bold
7	Joshbrush	Scripts	Semi-bold
8	Just Marsha	Scripts	Bold
9	Oldstyle	Serif	Semi-light
10	Bitstream Vera Mono	Freeform serifs	Semi-light
11	Duality	Slab serifs	Medium
12	Jura	Modern serifs	Medium
13	Elegan Tech	Sans Serif	Semi-light
14	Pilsen Plakat	Sans Serif	Semi-light
15	Planer	Sans Serif	Semi-light
16	Schul Vokal	Sans Serif	Semi-light
17	Sling	Sans Serif	Semi-light
18	Champagne Limousines	Sans Serif	Semi-light
19	Imperfecta	Sans Serif	Semi-light
20	Geo Sans Light	Sans Serif	Semi-light
21	Built Titling	Sans Serif	Semi-light
22	Comfortaa	Sans Serif	Semi-light
23	KG Why You Gotta Be So Mean	Sans Serif	Semi-light
24	Alpaca	Sans Serif	Semi-light
25	Qualio (regular)	Sans Serif	Semi-light
26	Qualio (italic)	Sans Serif	Semi-light

Tabel C.2 Font yang Digunakan pada Dataset (2)

No	Nama Font	Jenis Font	Ketebalan
27	Bebas (regular)	Sans Serif	Semi-light
28	Tall Dark and Handsome	Sans Serif	Semi-light
29	Calling Cards (regular)	Sans Serif	Semi-light
30	Stentiga	Sans Serif	Medium
31	Anorexia	Sans Serif	Medium
32	Francophil Sans	Sans Serif	Medium
33	Roboto (regular)	Sans Serif	Medium
34	Roboto (italic)	Sans Serif	Medium
35	Kenyan Coffee	Sans Serif	Medium
36	Bignoodletitling	Sans Serif	Medium
37	Libel Suit	Sans Serif	Medium
38	Coyote	Sans Serif	Medium
39	Calling Cards (bold)	Sans Serif	Medium
40	TGL 0-16	Sans Serif	Semi-bold
41	Romerio	Sans Serif	Semi-bold
42	Bison 2	Sans Serif	Semi-bold
43	Coolvetica	Sans Serif	Semi-bold
44	X-heighting	Sans Serif	Semi-bold
45	IJSkelder	Sans Serif	Semi-bold
46	Piximisa	Sans Serif	Bold
47	Steelfish	Sans Serif	Bold
48	School Times	Sans Serif	Bold
49	KG Let Her Go	Sans Serif	Bold
50	Kingthings Organica	Basic	Bold
51	Dealerplate California	Basic	Semi-light
52	Nokia Cellphone	Bitmap	Bold
53	SF Intoxicated Blue	Fancy	Semi-light
54	SF Arch Rival	Fancy	Semi-light
55	WBX Grannyt	Fancy	Semi-light
56	Jungle Land	Fancy	Medium
57	Go Bear	Fancy	Medium
58	Cozy Caps	Fancy	Medium
59	Komikaze	Fancy	Semi-bold
60	Trevor	Fancy	Semi-bold

BIODATA PENULIS

Penulis bernama Yolanda Hertita Pratama, putri pertama dari dua bersaudara yang lahir di Probolinggo pada tanggal 09 Maret 1998. Penulis telah menjalani masa pendidikan di Madrasah Aliyah Negeri 3 Malang pada tahun 2013 hingga 2016. Pada masa penulisan, penulis sedang menempuh masa studi S1 semester ketujuh di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember.

Selama masa studi S1, penulis memiliki ketertarikan yang dalam mengenai *Artificial Intelligence*, *Deep Learning*, dan *Natural Language Processing*. Penulis pernah menjadi asisten dosen pada mata kuliah Dasar Pemrograman, Matematika Informatika, Probabilistik dan Statistika, dan Kecerdasan Buatan. Dalam mengembangkan kemampuan yang dimiliki, penulis aktif melakukan riset maupun mengikuti kompetisi di bidang Data Mining. Prestasi yang pernah dicapai penulis yaitu menjadi juara 1 pada kompetisi pemrograman Techphoria 2018 dan finalis pada Gemastik 2017 dan 2018 pada bidang Data Mining. Penulis juga memiliki pengalaman magang di PT Hijup.com Jakarta.

Di luar kesibukan akademik, penulis cukup aktif berorganisasi di dalam kampus dengan menjadi wakil kepala departemen Teknologi Himpunan Mahasiswa Teknik Computer-Informatika 2019/2020. Penulis juga pernah membantu dalam kepanitiaan dalam jurusan, yaitu staff *National Programming Contest* Schematics 2017-2018.