Final Exam

Geoffrey Ulman
CSI873

December 2011

1 Implementation

The support vector machine constrained optimization problem was modeled and solved using AMPL (`www.ampl.com/`). Computations were performed using the LOQO solver (`www.princeton.edu/~rvdb/loqo/LOQO.html`) on the NEOS server (`www.neos-server.org/neos/solvers/`). Once the solution to the dual problem was obtained, the α values were imported into Java.

Java (version 1.6.0_27) was used to make the final classification. The code is available as a Subversion repository on Google Code at `http://code.google.com/p/csi873/`. Compiling and running the code requires the Java build tool Maven (`http://maven.apache.org/`).

2 Model

The following AMPL model was used to solve the dual SVM problem in Equation 1 with a polynomial kernel.

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{l} \alpha_i - 0.5 \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j K\left(x_i, x_j\right) \\ s.t. \\ \sum_{i=1}^{l} \alpha_i y_i = 0 \\ 0 \geq \alpha_i \geq C, i = 1, 2, ..., l \end{aligned} \tag{1}$$

```
model;

# number of training examples
param l;

# number of input parameters (pixels in digit image)
```

```
param n;

# weight on xi penalty coefficient in primal problem
param C;

# parameters for polynomial machine kernel
param alpha;
param beta;
param delta;

# output vector (1 or -1)
param y { 1..l };

# input data
param x { 1..l, 1..n };

# dual problem variables and simple constraints
var a {1..l} >= 0, <= C;

maximize obj: sum { i in 1..l } a[i] - 0.5 *
              sum { i in 1..l, j in 1..l } a[i] * a[j] * y[i] * y[j] *
              ( alpha * ( sum { k in 1..n } x[i,k] * x[j,k] ) + beta ) ^ delta;

s.t. const: sum { i in 1..l } a[i] * y[i] = 0;

option solver loqo;
```

The model used for the radial basis kernel was almost identical except for the objective (and some different parameters):

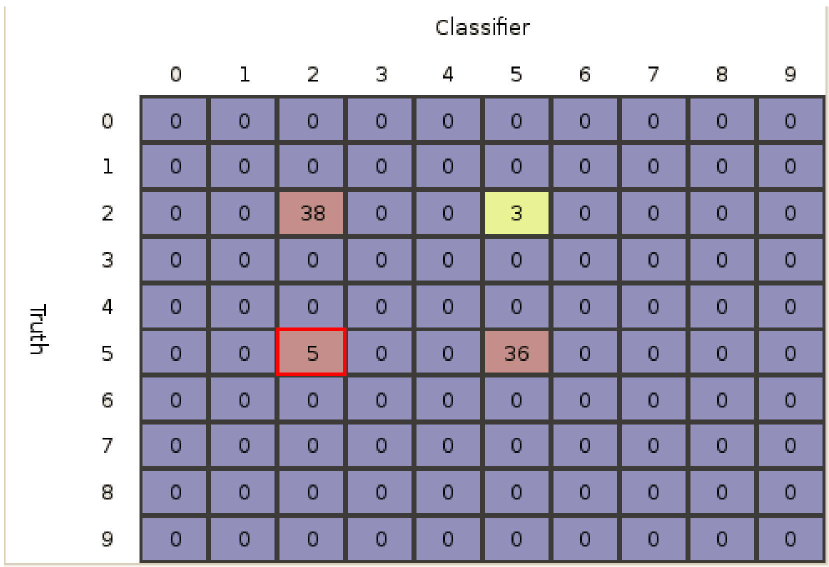
```
# parameters for radial basis function kernel
param gamma;

maximize obj: sum { i in 1..l } a[i] - 0.5 *
              sum { i in 1..l, j in 1..l } a[i] * a[j] * y[i] * y[j] *
              exp( -gamma * ( sum { k in i..n } ( x[i,k] - x[j,k] )^2 ) );
```

3 Two Digit Results

When the model was trained with the digit "2" and digit "5" training data, a misclassification error (on the testing data set) of 0.098 was achieved with the polynomial kernel and 0.110 with the radial kernel. The full results are summarized in Table 1.

Table 1: Digit 2 vs 5 Error

Data Set	Error	95% Confidence Interval	
		Lower Bound	Upper Bound
Polynomial Training	0.000	0.000	0.000
Polynomial Testing	0.098	0.033	0.162
Radial Training	0.027	0.004	0.050
Radial Testing	0.110	0.042	0.177



Truth \ Classifier	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	38	0	0	3	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	5	0	0	36	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Figure 1: Polynomial Kernel 2 Digit Test Error Confusion Matrix

Figures 1 and 2 show the confusion matrices for the polynomial and radial kernels respectively. Figures 3, 4, 5, and 6 display images for the correctly classified training data samples for the polynomial and radial kernels applied to the ”2“ versus ”5“ classification problem. Figures 7 and 8 display the incorrectly classified ”2“ digits for the polynomial and radial kernels. Figures 9 and 10 display the incorrectly classified ”5“ digits for the polynomial and radial kernels.

Based on these experiments, the polynomial kernel was chosen to be used for the full problem. However, both methods performed quite well and their 95% confidence intervals have significant overlap. So it is unclear which method is actually superior for this handwriting problem.

Truth \ Classifier	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	38	0	0	3	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	6	0	0	35	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Figure 2: Radial Kernel 2 Digit Test Error Confusion Matrix

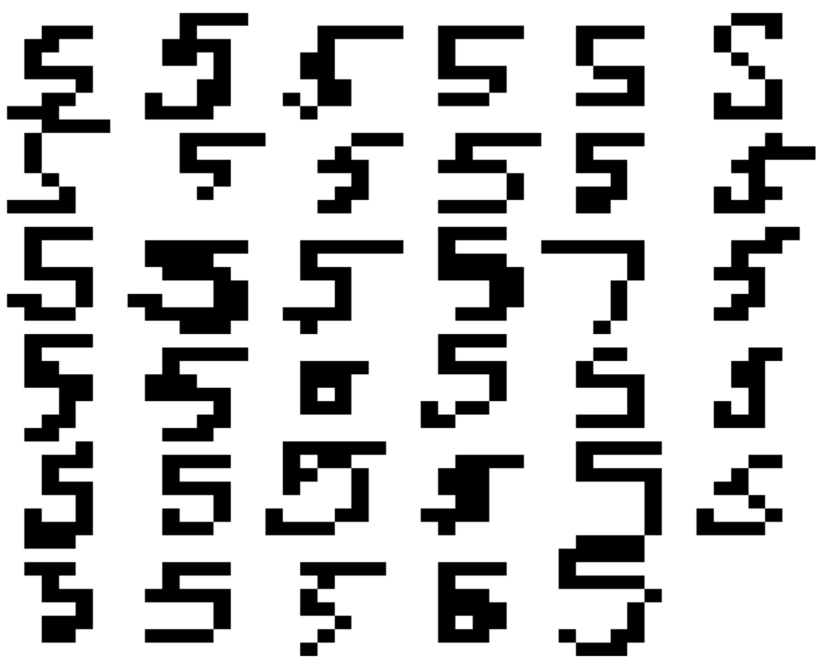


Figure 3: Radial Kernel Correctly Classified 5 Digits

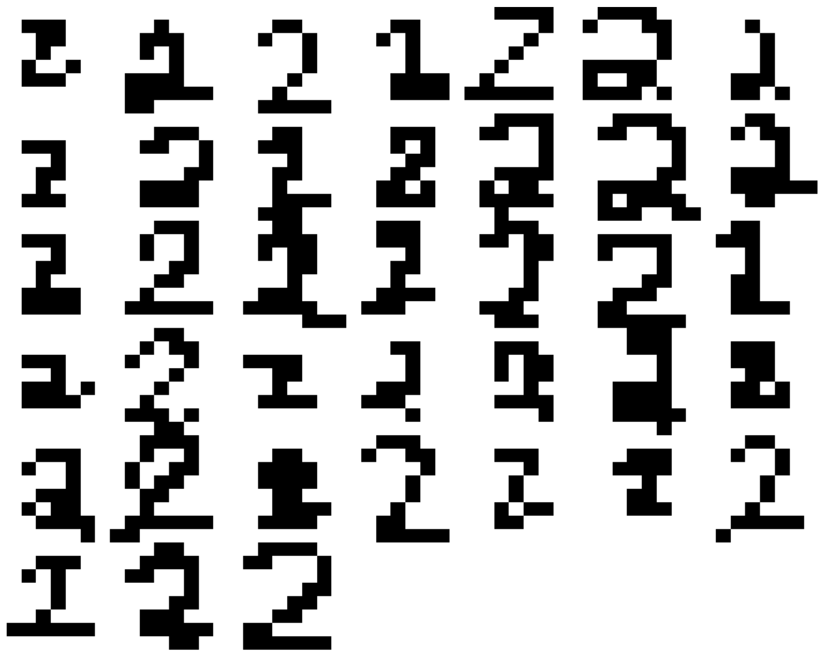


Figure 4: Radial Kernel Correctly Classified 2 Digits

4 All Digit Results

A misclassification error (on the full ten digit testing data set) of 0.200 was achieved with the polynomial kernel. The full results are summarized in Table 2. Figures 11 and 12 provide confusion matrices for the testing and training data sets.

5 Parameterization

The α_i upper bound parameter C was initially set at 100. Table 3 gives the number of support vectors (with non-zero and non-C α_i value) for each of the

Table 2: All Digits Error

Data Set	Error	95% Confidence Interval	
		Lower Bound	Upper Bound
Polynomial Training	0.029	0.018	0.040
Polynomial Testing	0.200	0.161	0.239

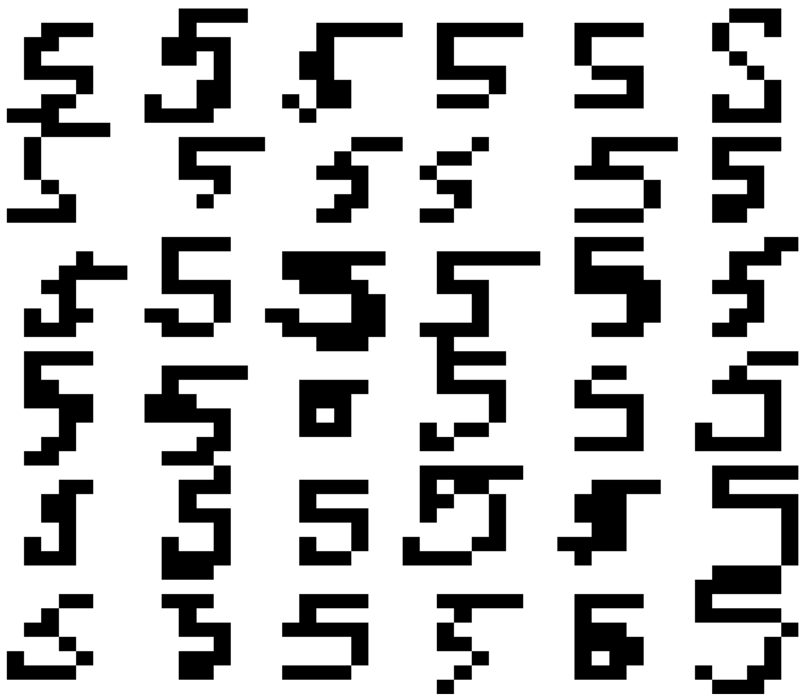


Figure 5: Polynomial Kernel Correctly Classified 5 Digits

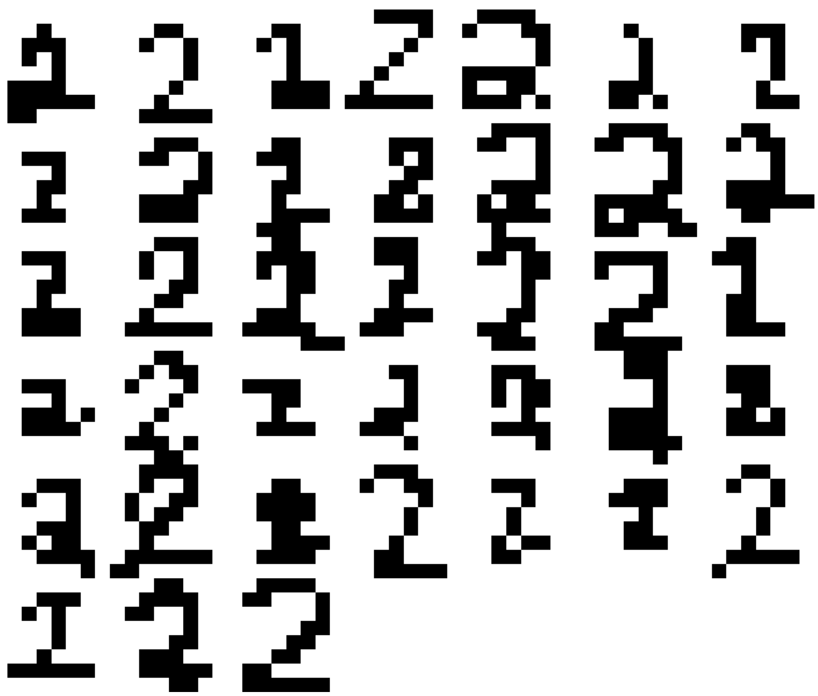


Figure 6: Polynomial Kernel Correctly Classified 2 Digits

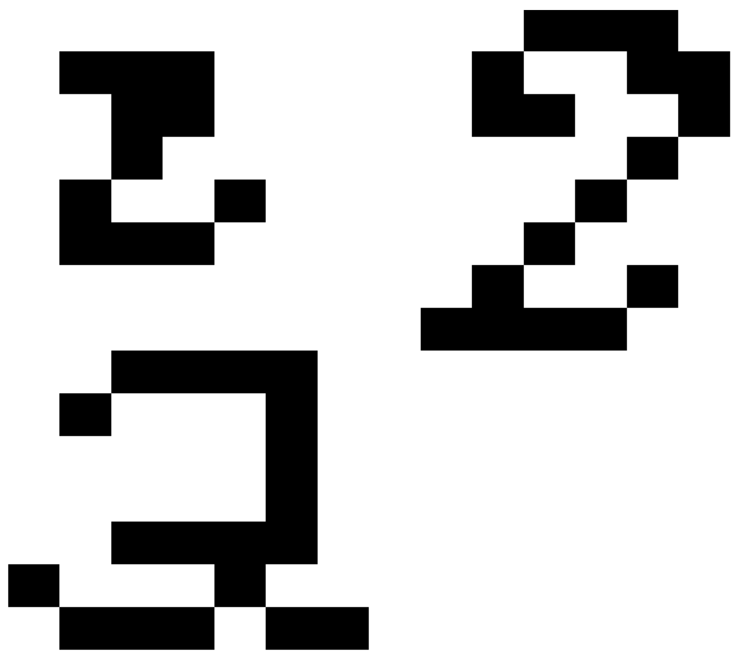


Figure 7: Polynomial Kernel 2 Misclassified as 5

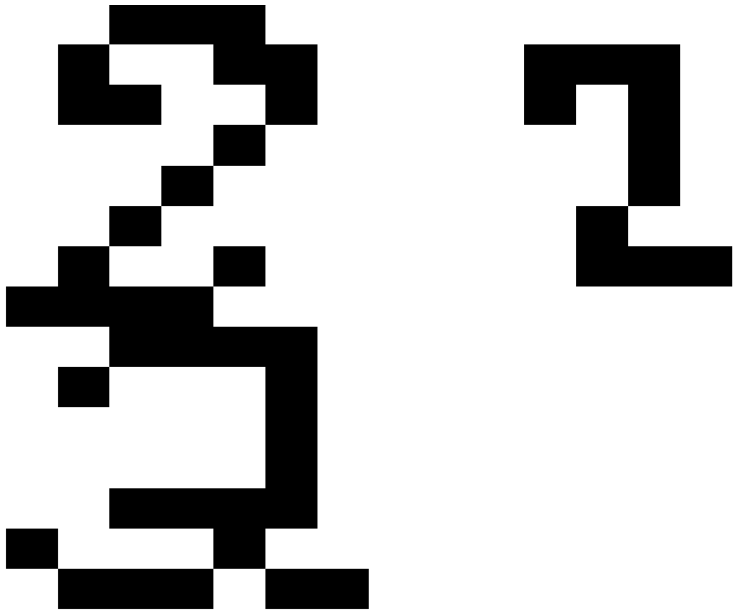


Figure 8: Radial Kernel 2 Misclassified as 5

10 SVM optimization problems solved for the 10 digit case. Out of 930 input vectors, only about 10% to 20% are chosen as support vectors by the solver. This relatively low number of support vectors indicates that the choice of C as 100 was reasonable.

6 Comparison

Figure 4 provides an error rate comparison between the four major classification methods which were applied to the handwriting data set. Weighted K-Nearest Neighbors with $k = 7$ came close to the performance achieved by the polynomial kernel support vector machine (with 0.217 error versus 0.200 for the support vector machine). Naive Bayes and Neural Networks trailed with 0.388 and 0.463 misclassification error respectively.

7 Appendix

The appendix to this report contains a number of supporting documents. The Java source code used to read the AMPL results, perform the classification, and compute statistics is included first. The code is also available at `http://code.google.com/p/csi873/source/browse/trunk/src/main/java/edu/gmu/classifier/svm/ampl/DataFileGenerator.java`.

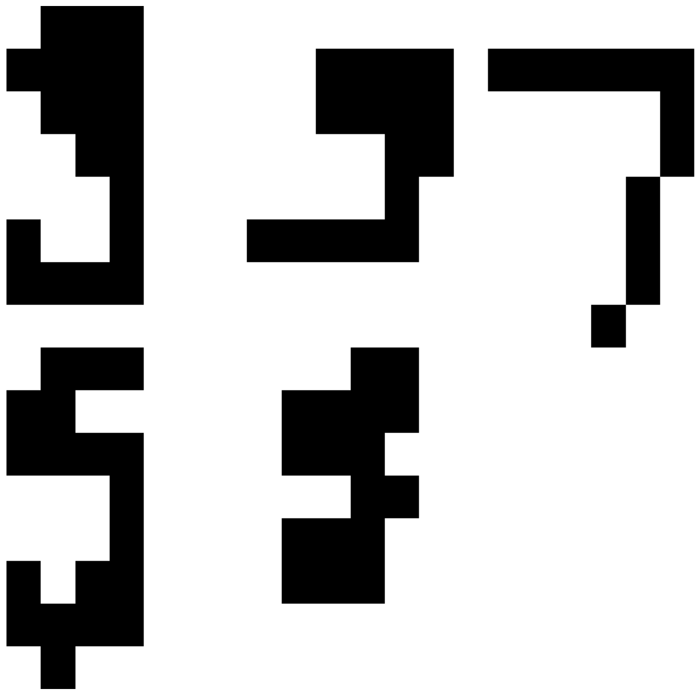


Figure 9: Polynomial Kernel 5 Misclassified as 2

Table 3: Number of Support Vectors Out Of 930

Digit	Support Vector Count
0	150
1	121
2	174
3	189
4	145
5	216
6	170
7	147
8	187
9	155

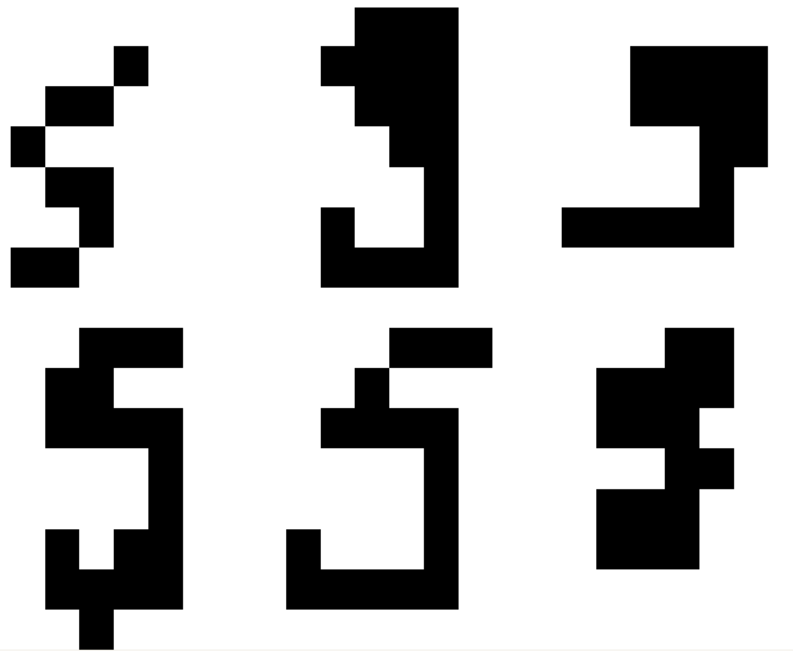


Figure 10: Radial Kernel 5 Misclassified as 2

Classifier

Truth	0	1	2	3	4	5	6	7	8	9
0	37	0	0	0	2	1	1	0	0	0
1	0	36	1	2	0	0	1	1	0	0
2	0	2	31	2	0	0	0	3	3	0
3	0	2	2	33	0	1	1	1	1	0
4	0	1	0	0	32	0	2	2	1	3
5	1	1	1	1	1	31	1	3	1	0
6	0	1	1	0	3	0	35	0	0	1
7	0	2	0	0	2	0	0	36	1	0
8	0	0	0	0	1	4	0	1	30	5
9	0	1	0	1	4	1	0	7	0	27

Figure 11: Polynomial Kernel 10 Digit Test Error Confusion Matrix

Truth \ Classifier	0	1	2	3	4	5	6	7	8	9
0	93	0	0	0	0	0	0	0	0	0
1	0	93	0	0	0	0	0	0	0	0
2	0	0	91	1	1	0	0	0	0	0
3	0	0	2	89	0	1	0	1	0	0
4	0	0	0	0	89	0	1	0	0	3
5	0	0	0	0	0	91	2	0	0	0
6	0	1	0	0	0	1	91	0	0	0
7	0	0	0	0	0	0	0	93	0	0
8	0	0	2	0	0	0	2	0	88	1
9	0	2	0	0	4	0	0	2	0	85

Figure 12: Polynomial Kernel 10 Digit Training Error Confusion Matrix

Table 4: Missclassification Error Overview

Algorithm	Error	95% Confidence Interval	
		Lower Bound	Upper Bound
Support Vector Machine	0.200	0.161	0.239
K-Nearest Neighbors	0.217	0.177	0.257
Naive Bayes	0.388	0.341	0.435
Neural Network	0.463	0.415	0.512

Following the Java code are the AMPL model files (both with the polynomial and radial kernels) and the AMPL data file for the ”2“ versus ”5“ classification problem (the data file for the full problem is omitted due to length). However, all AMPL model, data, and NEOS output files are also available at `http://code.google.com/p/csi873/source/browse/#svn%2Ftrunk%2Ffinal%2Fampl`.

References

[1] Tom M. Mitchell, *Machine Learning*, WCB McGraw-Hill, Boston, 1997.

```java
package edu.gmu.classifier.svm.ampl;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import edu.gmu.classifier.database.ResultsUploader;
import edu.gmu.classifier.database.UploadResultQuery;
import edu.gmu.classifier.database.UploadRunQuery;
import edu.gmu.classifier.io.DataLoader;
import edu.gmu.classifier.io.TrainingExample;

public class DataFileGenerator
{
	// a functor interface which defines a function for calculating the
	// y (output) value for a given training example
	// this function should always return either 1 or -1
	public interface OutputGenerator
	{
		public double getOutput( TrainingExample data );
	}

	// a functor interface which defines a function that takes
	// two input vectors (two 64 length vectors containing 0 or 1
	// in each element representing a handwriting sample) and outputs
	// a scalar value.
	public interface Kernel
	{
		public double getValue( double[] x1, double[] x2 );
	}

	// the polynomial kernel
	public static class Polynomial implements Kernel
	{
		double alpha, beta, delta;

		public Polynomial( double alpha, double beta, double delta )
		{
			this.alpha = alpha;
			this.beta = beta;
			this.delta = delta;
		}

		@Override
		public double getValue( double[] x1, double[] x2 )
		{
			double dot = 0.0;
			for ( int i = 0; i < x1.length; i++ )
			{
				dot += x1[i] * x2[i];
			}

			return Math.pow( alpha * dot + beta, delta );
		}
	}

	// the radial basis kernel
	public static class Radial implements Kernel
	{
		double gamma;

		public Radial( double gamma )
		{
			this.gamma = gamma;
		}

		@Override
		public double getValue( double[] x1, double[] x2 )
		{
			double norm = 0.0;
			for ( int i = 0; i < x1.length; i++ )
			{
				norm += Math.pow( x1[i] - x2[i], 2.0 );
			}

			return Math.exp( -gamma * norm );
		}
	}
```

```java
	// The output generator for one digit versus all others.
	// If the TrainingExample is an instance of the digit
	// the result is 1.0 otherwise it is -1.0.
	public static class OneVersusAll implements OutputGenerator
	{
		protected int digit;

		public OneVersusAll( int digit )
		{
			this.digit = digit;
		}

		@Override
		public double getOutput( TrainingExample data )
		{
			return data.getDigit( ) == digit ? 1 : -1;
		}
	};

	// The output generator for the two class (one digit versus
	// one other digit) problem.
	// If the TrainingExample is an instance of digit1 the
	// result is a 1.0 otherwise it is -1.0.
	public static class TwoClass implements OutputGenerator
	{
		protected int digit1;
		protected int digit2;

		public TwoClass( int digit1, int digit2 )
		{
			this.digit1 = digit1;
			this.digit2 = digit2;
		}

		@Override
		public double getOutput( TrainingExample data )
		{
			int digit = data.getDigit( );

			if ( digit == digit1 )
				return 1;
			else if ( digit == digit2 )
				return -1;
			else
				return 0;
		}
	};

	// a routine for generating AMPL data files from the provided training example data files
	// this generates 11 data files (ten for the 10 digit classification problem and one for
	// the 2 versus 5 classification problem).
	public static void generateAllDataFiles( String inDirectoryString, String outDirectoryString ) throws IOException
	{
		generateDataFile( inDirectoryString, outDirectoryString, "classify_2-5", 2, 5 );

		for ( int i = 0; i < 10; i++ )
		{
			generateDataFile( inDirectoryString, outDirectoryString, String.format( "classify_%d", i ), i );
		}
	}

	// a helper routine which generates a single AMPL data file using
	// data from all the digits and classifying the given digit against all others
	public static void generateDataFile( String inFileName, String outDirectoryName, String outFilePrefix, int digit ) throws
IOException
	{
		List<TrainingExample> dataList = DataLoader.loadDirectoryTrain( inFileName );

		File outDirectory = new File( outDirectoryName );
		File outFile = new File( outDirectory, outFilePrefix + ".dat" );
		outputDataFile( new FileOutputStream( outFile ), dataList, new OneVersusAll( digit ) );
	}

	// a helper routine which generates a single AMPL data file using
	// data from only the two provided digits
	public static void generateDataFile( String inFileName, String outDirectoryName, String outFilePrefix, int digit1, int digit2 )
throws IOException
	{
		List<TrainingExample> dataList = loadData( inFileName, false, digit1, digit2 );

		File outDirectory = new File( outDirectoryName );
		File outFile = new File( outDirectory, outFilePrefix + ".dat" );
		outputDataFile( new FileOutputStream( outFile ), dataList, new TwoClass( digit1, digit2 ) );
	}

	// loads the training examples corresponding to the two given digits from either the test or training data set
	public static List<TrainingExample> loadData( String inFileName, boolean test, int digit1, int digit2 ) throws IOException
	{
		List<TrainingExample> dataList = test ? DataLoader.loadDirectoryTest( inFileName ) : DataLoader.loadDirectoryTrain
```

```
( inFileName );

                List<TrainingExample> filteredList = new ArrayList<TrainingExample>( dataList.size( ) );
                for ( TrainingExample data : dataList )
                {
                        if ( data.getDigit( ) == digit1 || data.getDigit( ) == digit2 )
                        {
                                filteredList.add( data );
                        }
                }

                return filteredList;
        }

        // generates an AMLP data file for the given dataList and output generator
        public static void outputDataFile( OutputStream stream, List<TrainingExample> dataList, OutputGenerator gen ) throws IOException
        {
                BufferedWriter out = new BufferedWriter( new OutputStreamWriter( stream ) );

                out.write( "data;" );
                out.newLine( );

                int l = dataList.size( );
                out.write( String.format( "param l := %d;%n", l ) );

                int n = dataList.get( 0 ).getInputs( ).length;
                out.write( String.format( "param n := %d;%n", n ) );

                out.write( String.format( "param C := %f;%n", 100.0 ) );

                // Radial Kernel Parameters
                //out.write( String.format( "param gamma := %f;%n", 0.0521 ) );

                // Polynomial Kernel Parameters
                out.write( String.format( "param alpha := %f;%n", 0.0156 ) );

                out.write( String.format( "param beta := %f;%n", 0.0 ) );

                out.write( String.format( "param delta := %f;%n", 3.0 ) );

                out.write( String.format( "param y :=%n" ) );
                for ( int i = 0; i < l; i++ )
                {
                        TrainingExample data = dataList.get( i );
                        out.write( String.format( " %d %.1f%n", i + 1, gen.getOutput( data ) ) );
                }
                out.write( ";" );
                out.newLine( );

                out.write( String.format( "param x:" ) );
                for ( int i = 0; i < n; i++ )
                {
                        out.write( String.format( " %d", i + 1 ) );
                }
                out.write( String.format( " :=%n" ) );
                for ( int i = 0; i < l; i++ )
                {
                        TrainingExample data = dataList.get( i );
                        double[] input = data.getInputs( );

                        out.write( String.format( " %d", i + 1 ) );

                        for ( int j = 0; j < n; j++ )
                        {
                                out.write( String.format( " %.1f", input[j] ) );
                        }

                        out.newLine( );
                }
                out.write( ";" );
                out.newLine( );

                out.close( );
        }

        // reads a NEOS AMPL output file and returns the calculated alpha values
        public static double[] read_a( String file ) throws IOException
        {
                FileInputStream stream = new FileInputStream( file );
                try
                {
                        return read_a( stream );
                }
                finally
                {
                        stream.close( );
                }
        }

        // reads a NEOS AMPL output file and returns the calculated alpha values
```

```
	public static double[] read_a( InputStream stream ) throws IOException
	{
		List<Double> list = new ArrayList<Double>( );

		BufferedReader in = new BufferedReader( new InputStreamReader( stream ) );
		String line = null;
		boolean parseMode = false;

		while ( ( line = in.readLine( ) ) != null )
		{
			if ( parseMode )
			{
				String[] tokens = line.trim( ).split( "[\\s]+" );

				try
				{

					for ( int i = 0; i < tokens.length / 2; i++ )
					{
						int index = Integer.parseInt( tokens[i * 2] ) - 1;
						double value = Double.parseDouble( tokens[i * 2 + 1] );

						ensureLength( index, list );

						list.set( index, value );
					}

				}
				catch ( NumberFormatException e )
				{
					parseMode = false;
				}
			}
			else if ( line.startsWith( "a [*] :=" ) )
			{
				parseMode = true;
			}
		}

		double[] array = new double[list.size( )];
		for ( int i = 0; i < list.size( ); i++ )
			array[i] = list.get( i );

		return array;
	}

	// calculate b's for each a
	// only one is needed for an i s.t. 0 < a[i] < C, but this is a good check
	public static double[] calculate_b( List<TrainingExample> dataList, OutputGenerator out, Kernel kernel, double C, double[] a )
throws IOException
	{
		double[] b = new double[a.length];

		for ( int i = 0; i < b.length; i++ )
		{
			double[] x_i = dataList.get( i ).getInputs( );
			double y_i = out.getOutput( dataList.get( i ) );

			double sum = 0.0;
			for ( int j = 0; j < b.length; j++ )
			{
				TrainingExample x = dataList.get( j );
				double[] x_j = x.getInputs( );
				double y_j = out.getOutput( x );

				sum += y_j * a[j] * kernel.getValue( x_j, x_i );
			}

			b[i] = sum - y_i;
		}

		return b;
	}

	/**
	 * Makes a classification decision for the 2 versus 5 case based on the AMPL solution.
	 *
	 * @param dataListTest a list of data samples of classify
	 * @param dataListTrain the training data list used to train the svm classifier
	 * @param out a generator for calculating expected output values from the training data
	 * @param kernel the kernel used in the AMPL model to calculate the alpha vector
	 * @param a the alpha vector generated via AMPL
	 * @param b the beta value calculated from the AMPL solution
	 * @return a vector containing the predicted y values for each testing example
	 */
	public static double[] calculate_y_predicted( List<TrainingExample> dataListTest, List<TrainingExample> dataListTrain,
OutputGenerator out, Kernel kernel, double[] a, double b )
	{
		double[] y_predicted = new double[ dataListTest.size( ) ];
```

```
		// iterate over the training examples
		for ( int i = 0; i < dataListTest.size( ); i++ )
		{
			TrainingExample x_i = dataListTest.get( i );

			// apply the formula from the svm slides to compute a y_predicted value
			// based on the alpha vector (solution to the dual problem)
			double sum = 0.0;
			for ( int j = 0; j < a.length; j++ )
			{
				TrainingExample x_j = dataListTrain.get( j );
				double y_j = out.getOutput( x_j );
				double a_j = a[j];

				sum += y_j * a_j * kernel.getValue( x_j.getInputs( ), x_i.getInputs( ) );
			}

			y_predicted[i] = sum - b;
		}

		return y_predicted;
	}

	// ensures that the length of the provided list is at least large enough to contain index
	private static void ensureLength( int index, List<Double> list )
	{
		if ( list.size( ) > index ) return;

		for ( int i = list.size( ); i <= index; i++ )
		{
			list.add( i, 0.0 );
		}
	}

	// uses calculate_y_predicted( ) to classify each testing example and compute an error rate
	public static int[] calculateErrorRate( List<TrainingExample> dataListTest, List<TrainingExample> dataListTrain,
OutputGenerator out, Kernel kernel, double[] a, double b )
	{
		int[] digit = new int[ dataListTest.size( ) ];

		double[] y = calculate_y_predicted( dataListTest, dataListTrain, out, kernel, a, b );

		double count = 0.0;
		for ( int i = 0; i < dataListTest.size( ); i++ )
		{
			double value = y[i];
			double predicted = y[i] > 0 ? 1.0 : -1.0;
			double actual = out.getOutput( dataListTest.get( i ) );

			digit[i] = predicted > 0 ? 2 : 5;

			if ( predicted == actual )
			{
				count += 1.0;
			}

			System.out.printf( "Value %.4f Predicted %.1f Actual %.1f%n", value, predicted, actual );
		}

		double errorRate = 1.0 - ( count / dataListTest.size( ) );
		double errorInterval = 1.96 * Math.sqrt( errorRate * ( 1 - errorRate ) / dataListTest.size( ) );

		System.out.printf( "Error Rate: %.3f Train Interval: (%.3f, %.3f)%n", errorRate, errorRate - errorInterval, errorRate +
errorInterval );

		return digit;
	}

	// a database helper method for uploading results in the SQL data format
	// required by the CSI710 handwriting sample viewer (used for generating
	// confusion matrices and handwriting sample visualizations)
	public static void uploadResultsTest2_5( String description, List<TrainingExample> list, int[] predicted )
	{
		int first2id = 171257;
		int first5id = 171380;

		UploadRunQuery uploadRunQuery = new UploadRunQuery( description, System.currentTimeMillis( ) );
		uploadRunQuery.runQuery( );
		int ixRunId = uploadRunQuery.getRunId( );

		int count2 = 0;
		int count5 = 0;

		for ( int i = 0 ; i < list.size( ); i++ )
		{
			TrainingExample data = list.get( i );
			String sClassification = String.valueOf( predicted[i] );

			int index;
			if ( data.getDigit( ) == 2 )
```

```
			{
				index = first2id + count2;
				count2 += 1;
			}
			else
			{
				index = first5id + count5;
				count5 += 1;
			}

			UploadResultQuery uploadResultQuery = new UploadResultQuery( index, ixRunId, sClassification );
			uploadResultQuery.runQuery( );
		}
	}

	// helper method for generating AMPL model and data files
	public static void generateAmplDataFiles( ) throws IOException
	{
		String inputDirectory = "/home/ulman/CSI873/midterm/data";
		String outputDirectory = "/home/ulman/CSI873/midterm/repository/final/ampl";
		generateAllDataFiles( inputDirectory, outputDirectory );
	}

	public static void generateTestingResultsPolynomial_25( ) throws IOException
	{
		generateTestingResults( new Polynomial( 0.0156, 0.0, 3.0 ), "SVM 2vs5 run α = 0.0156, β = 0, d = 3" );
	}

	public static void generateTestingResultsRadial_25( ) throws IOException
	{
		generateTestingResults( new Radial( 0.0521 ), "SVM 2vs5 radial" );
	}

	// runs two digit 2-5 classification problem and calculates and displays results
	public static void generateTestingResults( Kernel kernel, String name  ) throws IOException
	{
		List<TrainingExample> dataListTrain = loadData( "/home/ulman/CSI873/midterm/data", false, 2, 5 );
		List<TrainingExample> dataListTest = loadData( "/home/ulman/CSI873/midterm/data", true, 2, 5 );
		String outputDirectory = "/home/ulman/CSI873/midterm/repository/final/ampl";
		String temporaryOutput = String.format( "%s/%s", outputDirectory, "out.tmp" );

		double C = 100.0;
		OutputGenerator out = new TwoClass( 2, 5 );

		double[] a = read_a( temporaryOutput );
		double[] b = calculate_b( dataListTrain, out, kernel, C, a );

		double count = 0.0;
		double b_sum = 0.0;

		for ( int i = 0 ; i < a.length ; i++ )
		{
			if ( a[i] < C && a[i] > 0.001 )
			{
				System.out.printf( "%.4f %.12f%n", a[i], b[i] );
				b_sum += b[i];
				count += 1.0;
			}
		}

		double b_avg = b_sum / count;

		System.out.println( "Error rate on Training Data." );
		calculateErrorRate( dataListTrain, dataListTrain, out, kernel, a, b_avg );

		System.out.println( "Error rate on Testing Data." );
		int[] testPreditions = calculateErrorRate( dataListTest, dataListTrain, out, kernel, a, b_avg );

		uploadResultsTest2_5( name, dataListTest, testPreditions );
	}

	//////////////////////////////
	///  Full 10 Digit Problem ///
	//////////////////////////////

	// a helper data structure for storing the alpha output values from AMPL
	// along with the calculated b value and an OutputGenerator
	public static class Model
	{
		double[] a;
		double b;
		OutputGenerator out;

		public Model( double[] a, double b, OutputGenerator out )
		{
			this.a = a;
			this.b = b;
			this.out = out;
		}
	}
```

```
	// runs ten digit classification problem and calculates and displays results
	public static void generateTestingResultsAll( ) throws IOException
	{
		List<TrainingExample> dataListTrain = DataLoader.loadDirectoryTrain( "/home/ulman/CSI873/midterm/data" );
		List<TrainingExample> dataListTest = DataLoader.loadDirectoryTest( "/home/ulman/CSI873/midterm/data" );

		Kernel kernel = new Polynomial( 0.0156, 0.0, 3.0 );

		Map<Integer,Model> map = new HashMap<Integer,Model>( );
		for ( int i = 0 ; i < 10 ; i++ )
		{
			String outputDirectory = "/home/ulman/CSI873/midterm/repository/final/ampl";
			String temporaryOutput = String.format( "%s/%s", outputDirectory, String.format( "out_%d.txt", i ) );

			double C = 100.0;
			OutputGenerator out = new OneVersusAll( i );

			double[] a = read_a( temporaryOutput );
			double[] b = calculate_b( dataListTrain, out, kernel, C, a );

			double count = 0.0;
			double b_sum = 0.0;

			for ( int j = 0 ; j < a.length ; j++ )
			{
				if ( a[j] < C && a[j] > 0.001 )
				{
					System.out.printf( "%.4f %.12f%n", a[j], b[j] );
					b_sum += b[j];
					count += 1.0;
				}
			}

			double b_avg = b_sum / count;

			map.put( i, new Model( a, b_avg, out ) );
		}

		System.out.println( "Error rate on Training Data." );
		int[] trainPreditions = calculateErrorRate( dataListTrain, dataListTrain, kernel, map );

		System.out.println( "Error rate on Testing Data." );
		int[] testPreditions = calculateErrorRate( dataListTest, dataListTrain, kernel, map );

		uploadResultsAllTrain( trainPreditions );
		uploadResultsAllTest( testPreditions );
	}

	// uses calculate_y_predicted( ) to classify each testing example and compute an error rate
	public static int[] calculateErrorRate( List<TrainingExample> dataListTest, List<TrainingExample> dataListTrain, Kernel kernel, 
Map<Integer,Model> map )
	{
		int[] predicted_digit = calculate_y_predicted( dataListTest, dataListTrain, kernel, map );

		double count = 0.0;
		for ( int i = 0; i < dataListTest.size( ); i++ )
		{
			TrainingExample data = dataListTest.get( i );

			if ( data.getDigit( ) == predicted_digit[i] )
			{
				count += 1.0;
			}
		}

		double errorRate = 1.0 - ( count / dataListTest.size( ) );
		double errorInterval = 1.96 * Math.sqrt( errorRate * ( 1 - errorRate ) / dataListTest.size( ) );

		System.out.printf( "Error Rate: %.3f Train Interval: (%.3f, %.3f)%n", errorRate, errorRate - errorInterval, errorRate + 
errorInterval );

		return predicted_digit;
	}

	/**
	 * Makes a classification decision for the 10 digit classification problem based on the solutions
	 * to the ten individual AMPL problems.
	 *
	 * @param dataListTest a list of data samples of classify
	 * @param dataListTrain the training data list used to train the svm classifier
	 * @param out a generator for calculating expected output values from the training data
	 * @param kernel the kernel used in the AMPL model to calculate the alpha vector
	 * @param map the alpha and beta values generated via AMPL for each of the ten separate SVMs constructed (one for each digit)
	 * @return a vector containing the predicted y values for each testing example
	 */
	public static int[] calculate_y_predicted( List<TrainingExample> dataListTest, List<TrainingExample> dataListTrain, Kernel 
kernel, Map<Integer,Model> map )
	{
		int[] predicted_digit = new int[ dataListTest.size( ) ];
```

```
		for ( int i = 0; i < dataListTest.size( ); i++ )
		{
			TrainingExample x_i = dataListTest.get( i );

			int best_digit = -1;
			double best_value = Double.NEGATIVE_INFINITY;

			for ( Entry<Integer,Model> entry : map.entrySet( ) )
			{
				int digit = entry.getKey( );
				Model model = entry.getValue( );
				double[] a = model.a;
				double b = model.b;
				OutputGenerator out = model.out;

				double sum = 0.0;
				for ( int j = 0; j < a.length; j++ )
				{
					TrainingExample x_j = dataListTrain.get( j );
					double y_j = out.getOutput( x_j );
					double a_j = a[j];

					sum += y_j * a_j * kernel.getValue( x_j.getInputs( ), x_i.getInputs( ) );
				}

				double value = sum - b;

				if ( value > best_value )
				{
					best_value = value;
					best_digit = digit;
				}
			}

			predicted_digit[i] = best_digit;
		}

		return predicted_digit;
	}

	public static void uploadResultsAllTest( int[] predicted_digit )
	{
		uploadResultsAll( "svm_testing_polynomial_all", ResultsUploader.IX_TEST_FIRST_INDEX, predicted_digit );
	}

	public static void uploadResultsAllTrain( int[] predicted_digit )
	{
		uploadResultsAll( "svm_training_polynomial_all", ResultsUploader.IX_TRAIN_FIRST_INDEX, predicted_digit );
	}

	public static void uploadResultsAll( String description, int firstDataId, int[] predicted_digit )
	{
		UploadRunQuery uploadRunQuery = new UploadRunQuery( description, System.currentTimeMillis( ) );
		uploadRunQuery.runQuery( );
		int ixRunId = uploadRunQuery.getRunId( );

		for ( int i = 0 ; i < predicted_digit.length; i++ )
		{
			String sClassification = String.valueOf( predicted_digit[i] );

			UploadResultQuery uploadResultQuery = new UploadResultQuery( firstDataId + i, ixRunId, sClassification );
			uploadResultQuery.runQuery( );
		}
	}

	public static void main( String[] args ) throws IOException
	{
		generateTestingResultsAll( );
	}
}
```

```
model;

# number of training examples
param l;

# number of input parameters (number of pixels in the handwriting digit images)
param n;

# weight on xi penalty coefficient in primal problem
param C;

# parameters for radial basis function kernel
param gamma;

# output vector (1 or -1)
param y { 1..l };

# input data
param x { 1..l, 1..n };

# dual problem variables and simple constraints
var a {1..l} >= 0, <= C;

maximize obj: sum { i in 1..l } a[i] - 0.5 * sum { i in 1..l, j in 1..l } a[i] * a[j] * y[i] * y[j] * exp( -gamma * ( sum { k in i..n }
( x[i,k] - x[j,k] )^2 ) );

s.t. const: sum { i in 1..l } a[i] * y[i] = 0;

option solver loqo;
```

```
model;

# number of training examples
param l;

# number of input parameters (number of pixels in the handwriting digit images)
param n;

# weight on xi penalty coefficient in primal problem
param C;

# parameters for polynomial machine kernel
param alpha;
param beta;
param delta;

# output vector (1 or -1)
param y { 1..l };

# input data
param x { 1..l, 1..n };

# dual problem variables and simple constraints
var a {1..l} >= 0, <= C;

maximize obj: sum { i in 1..l } a[i] - 0.5 * sum { i in 1..l, j in 1..l } a[i] * a[j] * y[i] * y[j] * ( alpha * ( sum { k in 1..n } x
[i,k] * x[j,k] ) + beta ) ^ delta;

s.t. const: sum { i in 1..l } a[i] * y[i] = 0;

option solver loqo;
```

```
data;

param l := 186;
param n := 64;
param C := 100.000000;
param alpha := 0.015600;
param beta := 0.000000;
param delta := 3.000000;

param y :=
 1 1.0
 2 1.0
 3 1.0
 4 1.0
 5 1.0
 6 1.0
 7 1.0
 8 1.0
 9 1.0
 10 1.0
 11 1.0
 12 1.0
 13 1.0
 14 1.0
 15 1.0
 16 1.0
 17 1.0
 18 1.0
 19 1.0
 20 1.0
 21 1.0
 22 1.0
 23 1.0
 24 1.0
 25 1.0
 26 1.0
 27 1.0
 28 1.0
 29 1.0
 30 1.0
 31 1.0
 32 1.0
 33 1.0
 34 1.0
 35 1.0
 36 1.0
 37 1.0
 38 1.0
 39 1.0
 40 1.0
 41 1.0
 42 1.0
 43 1.0
 44 1.0
 45 1.0
 46 1.0
 47 1.0
 48 1.0
 49 1.0
 50 1.0
 51 1.0
 52 1.0
 53 1.0
 54 1.0
 55 1.0
 56 1.0
 57 1.0
 58 1.0
 59 1.0
 60 1.0
 61 1.0
 62 1.0
 63 1.0
 64 1.0
 65 1.0
 66 1.0
 67 1.0
 68 1.0
 69 1.0
 70 1.0
 71 1.0
 72 1.0
 73 1.0
 74 1.0
 75 1.0
 76 1.0
 77 1.0
 78 1.0
 79 1.0
 80 1.0
```

81 1.0
82 1.0
83 1.0
84 1.0
85 1.0
86 1.0
87 1.0
88 1.0
89 1.0
90 1.0
91 1.0
92 1.0
93 1.0
94 -1.0
95 -1.0
96 -1.0
97 -1.0
98 -1.0
99 -1.0
100 -1.0
101 -1.0
102 -1.0
103 -1.0
104 -1.0
105 -1.0
106 -1.0
107 -1.0
108 -1.0
109 -1.0
110 -1.0
111 -1.0
112 -1.0
113 -1.0
114 -1.0
115 -1.0
116 -1.0
117 -1.0
118 -1.0
119 -1.0
120 -1.0
121 -1.0
122 -1.0
123 -1.0
124 -1.0
125 -1.0
126 -1.0
127 -1.0
128 -1.0
129 -1.0
130 -1.0
131 -1.0
132 -1.0
133 -1.0
134 -1.0
135 -1.0
136 -1.0
137 -1.0
138 -1.0
139 -1.0
140 -1.0
141 -1.0
142 -1.0
143 -1.0
144 -1.0
145 -1.0
146 -1.0
147 -1.0
148 -1.0
149 -1.0
150 -1.0
151 -1.0
152 -1.0
153 -1.0
154 -1.0
155 -1.0
156 -1.0
157 -1.0
158 -1.0
159 -1.0
160 -1.0
161 -1.0
162 -1.0
163 -1.0
164 -1.0
165 -1.0
166 -1.0
167 -1.0
168 -1.0
169 -1.0
170 -1.0

```
 171 -1.0
 172 -1.0
 173 -1.0
 174 -1.0
 175 -1.0
 176 -1.0
 177 -1.0
 178 -1.0
 179 -1.0
 180 -1.0
 181 -1.0
 182 -1.0
 183 -1.0
 184 -1.0
 185 -1.0
 186 -1.0
;

param x: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 :=
 1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 4 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
 5 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
 6 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 7 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
 8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 10 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 11 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 12 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0
 13 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
 14 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 15 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
 16 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 17 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 18 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 19 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
 20 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0
 21 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 22 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 23 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 24 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 25 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 26 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 27 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
 28 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 29 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 30 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 31 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 32 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 33 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 34 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 35 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
 36 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 37 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0
 38 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
 39 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0
 40 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 41 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 42 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 43 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 44 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 45 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 46 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
 47 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 48 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 49 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 50 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 51 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 52 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 53 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 54 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
 55 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0
 56 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0
 57 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
 58 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 59 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
 60 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 61 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 62 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
 63 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
 64 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 65 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 66 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
 67 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 68 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 69 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 70 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
 71 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
 72 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 73 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 74 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0
 75 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0
 76 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 77 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
 78 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
 79 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 80 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
```

```
 81 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
 82 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 83 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
 84 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 85 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 86 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 87 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 88 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
 89 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 90 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 91 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 92 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0
 93 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
 94 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 95 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 96 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 97 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 98 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 99 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 100 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
 101 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 102 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 103 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 104 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
 105 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
 106 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 107 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 108 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 109 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 110 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 111 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 112 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
 113 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
 114 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 115 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 116 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 117 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 118 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
 119 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 120 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 121 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
 122 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 123 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 124 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 125 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
```

126 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
127 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
128 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
129 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
130 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
131 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
132 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
133 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0
134 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
135 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
136 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
137 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
138 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
139 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
140 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
141 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
142 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
143 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
144 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
145 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
146 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
147 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
148 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
149 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0
150 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
151 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
152 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
153 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
154 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
155 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
156 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
157 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
158 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
159 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
160 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
161 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0
162 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
163 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
164 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
165 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
166 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
167 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
168 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
169 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0
0.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
170 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 1.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0

```
 171 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 172 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 173 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 174 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 175 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
 176 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
 177 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 178 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 179 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
 180 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0
 181 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
 182 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
 183 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 184 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
 185 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0
 186 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0
1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
;
```

Output sample from NEOS Server for full SVM problem: digit 0 versus digit 1-9

```
*************************************************************

   NEOS Server Version 5.0
   Job#     : 48781
   Password : eyVHlKkZ
   Solver   : nco:LOQO:AMPL
   Start    : 2011-12-03 20:17:45
   End      : 2011-12-03 20:18:53
   Host     : neos-1.chtc.wisc.edu

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
*************************************************************
Job 48781 sent to neos-1.chtc.wisc.edu
password: eyVHlKkZ
---------- Begin Solver Output -----------
Executing /opt/neos/Drivers/loqo-ampl/loqo-driver.py at time: 2011-12-03 20:17:45.496927
File exists
You are using the solver loqo.
Executing AMPL.
processing data.
processing commands.

930 variables, all nonlinear
1 constraint, all linear; 930 nonzeros
        1 equality constraint
1 nonlinear objective; 930 nonzeros.

LOQO 6.07: optimal solution (41 QP iterations, 121 evaluations)
primal objective 4720.34696
  dual objective 4720.346974
a [*] :=
  1   2.54104e-07   234  17.4712       467   4.87526e-08   700   5.87498e-08
  2   1.19626e-07   235  30.5848       468   2.03188e-08   701   2.21626e-08
  3   1.34122e-08   236   1.22409e-07  469   8.16445e-09   702   6.0253e-08
  4   3.98039       237   8.33865e-09  470   8.14434e-09   703   1.55225e-08
  5   9.71702e-09   238   1.57041e-08  471  19.2816        704   1.61523e-08
  6   2.58456e-06   239   9.67456e-09  472   3.90321e-08   705   2.06737e-08
  7   9.93516e-09   240   6.95511e-08  473   8.43092e-09   706   1.83124e-08
  8   1.18059e-08   241   4.81713e-09  474   2.68542e-08   707   8.23788e-09
  9  79.0103        242   4.52188e-09  475   5.75013e-08   708   2.76888e-08
 10  66.8627        243   3.97575e-09  476   1.47491e-08   709   3.30243e-08
 11  29.4992        244   1.39611e-08  477   1.46324e-08   710   3.63636
 12  95.701         245   6.16609e-09  478   1.74349e-08   711   3.36838e-07
 13  50.4826        246   2.20327e-08  479   1.30328e-08   712   0.883297
 14  29.1869        247   2.18117e-08  480   7.13772e-09   713   1.33965e-07
 15  63.0077        248   1.58383e-08  481   2.80275e-08   714   4.01726e-08
 16 100             249   2.39507e-08  482   2.74134e-08   715   2.59658e-08
 17  19.0734        250   2.65994e-08  483   2.2169e-08    716   4.67839e-08
 18  57.3959        251   2.51157e-08  484   4.15796e-08   717   2.95377e-08
 19  54.6249        252   5.32651e-08  485   8.23745       718  89.2605
 20  53.6528        253   3.17409e-08  486   2.73684e-08   719   2.52053e-08
 21 100             254   6.59136e-08  487   1.32297e-08   720   1.77329e-08
 22  41.421         255   1.9533e-08   488   2.45174e-08   721   1.20815e-08
 23  91.599         256   6.35262e-09  489   1.48766e-08   722   6.50441e-09
 24 100             257   9.72672e-09  490   1.52682e-08   723   1.35738e-08
 25 100             258   4.07421e-09  491   8.14434e-09   724   2.76213e-08
 26 100             259   6.37795e-09  492   2.45174e-08   725  18.7365
 27 100             260   7.83337e-09  493   5.57514e-08   726   2.92533e-08
 28 100             261   9.37859e-09  494   6.42128e-08   727   1.05137e-08
 29 100             262   4.4875       495   5.48359e-08   728   2.13431e-08
 30 100             263   7.23894e-09  496   1.20387e-08   729   2.49374e-08
 31  55.3495        264   8.59847e-09  497   2.6448e-07    730   1.44496e-08
 32  23.8046        265   7.93346e-09  498   1.0586e-08    731   4.62007
 33 100             266   1.42034e-08  499   1.70863e-08   732   8.96058
 34   0.468614      267   3.82261e-08  500   8.20964e-09   733   2.0332e-08
 35 100             268   4.15888e-08  501   2.13967e-08   734   1.562e-07
 36  32.8671        269   1.48333e-07  502   8.33854e-08   735   1.02088e-08
 37   4.89027e-09   270   1.01998e-08  503   6.4014e-09    736   1.93147e-08
 38 100             271   1.81366e-08  504   2.18686e-08   737   1.94745e-08
 39 100             272   2.72315e-08  505   5.35617e-09   738   1.74742e-08
 40   8.59328e-09   273   1.674e-08    506   2.96089e-09   739   3.31582e-08
 41   1.45215e-08   274   4.64615e-08  507   8.86299e-09   740  29.6416
 42   6.23803e-08   275   1.35814e-07  508   1.9907e-08    741   4.5877e-08
 43   6.94796e-09   276   2.64469e-08  509   1.14618e-08   742   1.91541e-08
 44   2.29794e-08   277   1.82098e-08  510   8.52856e-09   743   2.2468e-08
 45   1.69539e-08   278   4.9414e-08   511   9.37591e-09   744   2.08865e-08
 46  13.4704        279   4.79054e-08  512   1.26803e-08   745   1.83089e-08
 47   6.9851e-08    280   2.55699e-08  513   7.76705e-09   746   2.96616e-08
 48   1.1076e-08    281   1.20888e-08  514  12.306         747   1.48627e-08
 49  40.5431        282   5.12537e-09  515   1.61456e-07   748  11.4963
 50   3.54224e-08   283   1.26779e-08  516   1.9956e-08    749   2.18718e-08
 51  59.5613        284  23.3861       517   3.70136e-08   750   3.91503e-08
```

52	29.3258	285	1.29012e-08	518	2.48358e-08	751	4.61848
53	9.82969e-09	286	1.13853e-08	519	1.04419e-08	752	1.47398e-08
54	17.9755	287	1.16266e-08	520	9.62092e-09	753	8.59089e-09
55	3.54791e-08	288	2.05013e-08	521	1.11784e-08	754	6.05113e-09
56	1.95069e-08	289	1.55203e-08	522	1.21725e-08	755	1.54709e-08
57	6.87856e-09	290	2.7949e-08	523	1.24913e-08	756	1.69017e-08
58	100	291	1.33001e-08	524	1.1214e-08	757	41.2744
59	1.91574	292	64.6035	525	5.03505e-08	758	0.492657
60	100	293	1.00033e-08	526	1.28497e-08	759	40.6976
61	100	294	7.72382e-09	527	8.47442e-09	760	4.16779e-08
62	1.68637e-06	295	1.9299e-08	528	1.8415e-08	761	1.59142e-08
63	3.22921e-08	296	2.73438e-08	529	9.88068e-08	762	4.06883e-08
64	100	297	2.76609e-08	530	2.72134e-07	763	1.01051e-08
65	100	298	1.45769e-08	531	8.65954	764	1.26638e-08
66	4.84071e-07	299	3.46291e-08	532	10.2414	765	1.3033e-08
67	100	300	8.41646e-09	533	1.69039e-08	766	2.04937e-08
68	75.5297	301	7.85982e-08	534	14.7536	767	33.7332
69	100	302	83.4905	535	1.09699e-08	768	7.50062
70	6.32841e-09	303	2.57202e-08	536	1.00735e-08	769	1.36782e-07
71	4.0333e-09	304	1.64889e-08	537	34.4198	770	7.19452e-09
72	2.22391e-08	305	2.29576e-08	538	1.29148e-08	771	6.05113e-09
73	100	306	1.87963e-08	539	12.9091	772	100
74	1.55623e-08	307	1.94092e-08	540	1.84808e-08	773	75.6824
75	100	308	1.25786e-07	541	66.3396	774	2.87735e-08
76	1.41779e-07	309	2.40743e-08	542	32.4381	775	5.60758e-09
77	2.94677e-08	310	1.3949e-08	543	9.75831e-09	776	9.80621e-09
78	100	311	9.40763e-09	544	25.0778	777	6.79259e-09
79	100	312	1.30028e-08	545	1.94404e-08	778	7.57336e-09
80	2.51005e-08	313	7.94895e-09	546	2.32392e-08	779	1.07562e-08
81	53.0538	314	7.03787e-09	547	7.2701e-08	780	1.12161e-08
82	100	315	3.02551e-08	548	9.85037e-08	781	1.52235e-08
83	100	316	8.11268e-09	549	8.72629	782	2.41152e-08
84	2.0639e-08	317	7.96406e-09	550	9.23216e-08	783	41.0723
85	8.55713e-09	318	6.09149e-09	551	3.29632e-05	784	4.33366e-09
86	75.1753	319	7.30344e-09	552	1.97674e-08	785	4.34579e-09
87	72.3981	320	3.22578	553	16.1555	786	16.4371
88	4.50408e-08	321	1.32705e-08	554	5.7855e-08	787	5.53727e-09
89	88.6124	322	9.33881e-09	555	2.08133e-08	788	9.71642e-09
90	2.68238e-08	323	7.04953e-09	556	6.3892e-08	789	1.50534e-08
91	1.74891e-08	324	8.13147e-09	557	36.2011	790	3.95065e-09
92	29.4659	325	8.9976e-09	558	7.53669	791	1.33771e-08
93	6.40354	326	6.25153e-09	559	4.44468e-08	792	8.09887e-09
94	5.70093e-07	327	4.44801e-09	560	85.0317	793	5.83494e-09
95	1.18143e-07	328	7.13239e-09	561	14.6422	794	8.33865e-09
96	95.4334	329	1.86176e-08	562	5.64557	795	3.45885e-07
97	2.16e-08	330	8.88263e-09	563	2.39389e-08	796	3.48068e-08
98	2.65149e-08	331	2.98174e-08	564	44.4075	797	81.2418
99	1.94551e-08	332	1.26561e-08	565	8.52262e-09	798	4.32205e-09
100	4.96008e-08	333	7.81472e-09	566	5.30586e-09	799	5.70056e-09
101	4.92112e-08	334	6.62479e-09	567	8.83365e-09	800	3.55003e-09
102	4.93881e-08	335	1.21847e-08	568	9.88997e-09	801	3.53798e-09
103	1.39675e-07	336	1.23951e-08	569	9.90753e-08	802	6.86435e-09
104	4.9432e-08	337	0.818902	570	100	803	6.81077e-09
105	1.02334e-07	338	5.11663	571	46.4094	804	1.57265e-08
106	9.91209e-08	339	0.526704	572	3.92026e-08	805	1.97077e-08
107	1.21861e-07	340	6.35364e-08	573	87.6409	806	1.7724e-08
108	3.37421e-06	341	1.34441e-08	574	5.6656e-08	807	1.0398e-08
109	1.43245e-07	342	1.76907e-08	575	3.03604e-08	808	1.93508e-08
110	1.02334e-07	343	2.80312e-08	576	1.44952e-08	809	1.68536e-08
111	5.70093e-07	344	3.10516e-08	577	3.47822	810	1.42215e-08
112	1.45749e-07	345	2.0243e-08	578	49.9234	811	1.81596e-08
113	1.04659e-07	346	1.40558e-08	579	7.98937e-08	812	5.92549e-09
114	1.04659e-07	347	2.06604e-08	580	2.55771e-08	813	6.46076e-08
115	6.14558e-08	348	8.71565e-08	581	7.93228e-07	814	3.70729e-09
116	2.48369e-08	349	1.70372e-08	582	2.19256	815	3.01992e-09
117	5.9402e-08	350	5.93449e-09	583	9.00642e-08	816	6.0275e-09
118	100	351	1.58516e-08	584	8.78359e-07	817	6.41464e-09
119	100	352	4.11252e-08	585	2.23319e-08	818	5.1466e-09
120	5.20151e-07	353	1.51187e-08	586	8.94785	819	6.16063e-09
121	5.70093e-07	354	2.16536e-08	587	8.65846e-08	820	5.50829e-09
122	5.70093e-07	355	1.57554e-08	588	34.0518	821	5.1513e-09
123	5.20151e-07	356	8.50927e-09	589	1.49159e-07	822	1.94483e-08
124	1.02334e-07	357	9.71347e-09	590	27.8738	823	1.64351e-08
125	21.5335	358	1.1988e-08	591	36.5993	824	4.27739e-09
126	6.97257e-08	359	1.3502e-08	592	1.33488e-08	825	3.18412e-09
127	6.80954e-08	360	1.12485e-08	593	1.29629e-08	826	9.85511e-09
128	5.77438e-08	361	1.13474e-08	594	4.59915e-08	827	17.9557
129	8.08881e-09	362	1.64766e-08	595	8.48536e-06	828	7.47141e-09
130	1.25587e-08	363	5.28064e-08	596	1.18722e-08	829	1.28493e-07
131	2.05536e-07	364	40.6276	597	3.79201e-08	830	7.47376e-09
132	3.78553e-08	365	3.06479e-08	598	1.15166e-08	831	2.65965e-08
133	1.02334e-07	366	2.14085e-08	599	5.95047e-09	832	4.09571e-08
134	2.37836e-08	367	2.70805e-06	600	3.60763e-08	833	59.31
135	5.35017e-08	368	4.58223	601	2.41787	834	7.85893e-09
136	1.70918e-07	369	1.10234e-08	602	8.47396e-09	835	30.7255
137	2.20649e-08	370	1.00832e-08	603	8.33494e-09	836	4.37655e-07
138	1.60545e-08	371	2.94072e-08	604	2.23342e-08	837	2.15417e-08
139	1.66958e-08	372	9.01655	605	2.10729e-08	838	2.52157e-08
140	5.9075e-08	373	2.94384e-08	606	2.98421e-08	839	1.69806e-08
141	1.4319e-08	374	2.40798e-08	607	1.03788e-08	840	8.3501e-09

142	9.66486e-09	375	20.1633	608	2.48364e-08	841	7.95741e-06
143	1.02334e-07	376	2.41576	609	12.1347	842	2.32881e-08
144	100	377	21.9702	610	2.6101e-08	843	22.3696
145	5.53809e-08	378	2.03594e-08	611	2.56254e-08	844	22.1924
146	5.53809e-08	379	33.1247	612	1.17583e-08	845	31.7939
147	1.99422e-08	380	1.27199e-08	613	7.72954e-09	846	21.2986
148	1.03966e-08	381	1.02964e-08	614	21.8124	847	5.63127e-08
149	1.2861e-08	382	1.23401e-08	615	3.11913e-08	848	1.28492e-08
150	1.09542e-08	383	7.62784e-08	616	3.11907e-08	849	2.73101e-07
151	2.13299e-08	384	27.2171	617	9.25047e-09	850	4.7705e-08
152	1.92667e-07	385	1.37407e-08	618	2.36977e-08	851	38.0972
153	100	386	4.72696e-08	619	1.1672e-08	852	7.86202e-08
154	47.0661	387	8.05227e-08	620	2.50991e-08	853	1.62926e-08
155	4.4398e-08	388	2.42263e-08	621	2.38919e-08	854	2.53616e-08
156	1.87558e-07	389	2.47875e-08	622	2.3416e-08	855	2.49449e-08
157	5.20151e-07	390	8.86408e-08	623	1.13234e-08	856	2.51318e-08
158	5.20151e-07	391	1.62591e-08	624	1.05402e-07	857	1.34258e-08
159	1.69504e-07	392	2.81867e-08	625	2.582e-06	858	2.0295e-08
160	1.79558e-07	393	1.70432e-08	626	2.67456e-08	859	2.10016e-08
161	6.64538e-08	394	2.99264e-08	627	1.29527e-08	860	2.14264e-08
162	100	395	4.06366e-08	628	6.15296e-09	861	2.53905e-08
163	1.06591e-08	396	4.0852e-08	629	17.7125	862	2.78768e-08
164	1.87558e-07	397	100	630	2.27535e-08	863	20.2213
165	2.9957e-08	398	38.3001	631	2.02758e-08	864	1.10992e-07
166	1.35284e-08	399	65.4451	632	15.1093	865	6.31223e-08
167	1.19697e-07	400	3.54449e-08	633	3.24086e-08	866	2.53534e-08
168	1.76641e-07	401	3.8498e-08	634	15.9598	867	1.7201e-08
169	4.19078e-08	402	2.36009e-08	635	36.2095	868	3.09273e-08
170	4.40335e-08	403	1.66105e-08	636	3.31771	869	3.77642e-08
171	2.33274e-08	404	3.01124e-08	637	1.62809e-08	870	1.16611e-08
172	3.43441e-08	405	3.01124e-08	638	1.16284e-05	871	1.89849e-08
173	1.02334e-07	406	1.52561e-08	639	24.9877	872	1.53376e-08
174	1.02334e-07	407	1.01579e-08	640	27.2412	873	4.40436e-08
175	1.87558e-07	408	2.83542e-08	641	3.74161e-08	874	5.12404e-08
176	3.52827e-07	409	2.49312e-08	642	1.75216e-08	875	9.95763e-09
177	1.87999e-08	410	1.39885e-08	643	1.35339e-08	876	7.19758e-09
178	1.28175e-08	411	1.88709e-08	644	5.54892e-08	877	1.11478e-08
179	2.56711e-08	412	1.53606e-08	645	24.2709	878	1.18418e-08
180	4.64274e-08	413	9.64378e-09	646	3.40897e-08	879	1.02442e-08
181	1.87558e-07	414	1.07262e-08	647	2.31233e-08	880	9.12941e-09
182	5.70093e-07	415	3.05385e-08	648	13.371	881	1.7495e-08
183	100	416	1.73858e-08	649	2.00226e-08	882	2.53534e-08
184	5.70093e-07	417	1.73858e-08	650	16.5607	883	1.14716e-08
185	1.02334e-07	418	2.8581e-08	651	2.38337	884	5.54365
186	4.9432e-08	419	2.8581e-08	652	0.495867	885	1.68253e-08
187	1.25038e-08	420	7.31885e-09	653	5.51548e-08	886	7.66081e-09
188	7.68674e-09	421	1.60356e-08	654	4.77801e-08	887	2.13601e-08
189	2.08558e-08	422	1.20679e-08	655	3.50108e-08	888	1.43996e-08
190	1.63374e-08	423	2.14087e-08	656	2.51646e-08	889	3.57152e-08
191	1.09468e-08	424	7.37835e-09	657	2.51499e-08	890	6.01533e-08
192	3.04858	425	1.87619e-08	658	12.582	891	8.7674e-09
193	35.4882	426	8.53409e-09	659	33.3076	892	8.32414e-09
194	7.4284e-08	427	1.33366e-08	660	4.70003e-08	893	1.60036e-08
195	1.14531e-08	428	2.79618e-08	661	5.1817e-08	894	9.38604e-09
196	1.57079e-08	429	34.6565	662	19.6549	895	1.26229e-08
197	1.5211e-08	430	1.61318e-08	663	4.25922e-08	896	1.17725e-08
198	2.65985e-08	431	5.57306e-09	664	3.34334e-08	897	2.64163e-08
199	4.84294e-08	432	2.49845e-07	665	53.2169	898	4.10521e-08
200	1.36097e-08	433	2.78925e-08	666	3.26911e-08	899	11.5756
201	1.81948e-08	434	9.49286	667	8.20046e-08	900	1.04456e-07
202	8.11654e-08	435	8.40378e-08	668	5.77246e-08	901	1.86493e-08
203	5.27549e-08	436	2.64663e-08	669	8.09602e-08	902	2.93149e-08
204	2.31288e-08	437	2.22699e-08	670	1.6309e-08	903	2.15849e-08
205	2.32679e-08	438	1.30377e-08	671	1.06244e-08	904	3.19172e-08
206	4.40799	439	1.66787e-08	672	1.87626e-08	905	1.93123e-08
207	12.5803	440	1.35817e-08	673	3.06638e-08	906	7.62597e-08
208	2.36189e-08	441	2.33235e-08	674	3.19671e-08	907	7.22766e-09
209	1.72547e-08	442	1.1371e-08	675	2.76627e-08	908	9.02114e-09
210	100	443	1.05029e-08	676	2.39489e-08	909	4.02114e-08
211	2.07402e-08	444	2.37505e-08	677	8.30498e-08	910	1.7339e-08
212	1.44934e-08	445	6.64982e-08	678	1.85878e-08	911	1.89005e-08
213	62.9367	446	1.11484e-08	679	1.92596e-08	912	1.63775e-08
214	2.67088e-08	447	1.66349e-08	680	4.38342e-08	913	6.99876e-09
215	2.09632e-08	448	1.7268e-07	681	4.37393e-08	914	2.82903e-08
216	1.92502e-08	449	2.94979	682	4.52816e-08	915	5.02608e-08
217	18.466	450	1.055e-08	683	9.63688e-08	916	1.28352e-08
218	85.3915	451	1.92705e-08	684	6.1303e-08	917	1.55777e-08
219	48.0768	452	1.0629e-08	685	3.76734e-08	918	1.64948e-08
220	9.20391e-09	453	1.43087e-08	686	4.72628e-08	919	5.05662e-08
221	6.67695e-09	454	2.81716	687	3.27956	920	5.05662e-08
222	1.60233e-08	455	1.46738e-06	688	5.28699e-08	921	1.58743e-08
223	8.08041e-09	456	1.21587e-08	689	4.98319e-08	922	1.36395e-08
224	1.30634e-07	457	3.52278e-08	690	1.89477e-08	923	3.94626e-08
225	1.69851e-08	458	2.73913e-08	691	5.39992e-09	924	1.3901e-07
226	13.4811	459	5.78015e-08	692	6.6157e-09	925	2.36461e-08
227	21.847	460	6.97589e-08	693	2.23245e-08	926	32.7848
228	1.45237e-08	461	2.40659e-08	694	2.43192e-08	927	1.17259e-08
229	5.71778e-09	462	1.85802e-08	695	1.54787e-08	928	9.78166e-09
230	6.08565e-09	463	3.24946e-08	696	1.2899e-08	929	1.61469e-08
231	9.02116e-09	464	3.06425e-08	697	4.29067e-08	930	10.4218

232	6.01545e-09	465	14.7284	698	4.65553
233	1.36818e-07	466	4.40484e-08	699	25.3571

;