Comp528
Assignment 3
By Robert Johnson

December 3, 2017

Contents

1 Explanation of KKK algorithm

KKK encryption isnt a mathematical algorithm but rather a system that works off synchronizing two neural networks, it does this by creating two neural networks of NK size of weights ranging from -L to L, where N is the amount of weights inside a percept and K is the amount of perceptrons there are in the network, for example a network with N=5, K=2, L=1 would have 2 perceptrons with 5 weights each, with weight values ranging from 1 to -1.
The encryption is used by users A and B who create two random networks with parameters (N, K, L) these networks are hidden. Each user receives the same n random numbers which can be (-1 or 1) which are then pumped into the neural network, each network takes these inputs and applies them to the weights inside each percept creating a value which is individual to each percept using the following formula.
$sum - (neuralNet.weights[j] * inputs[j])$;
If $sum \leq 0$ after applying this formula to each weight then it should be bounded to 1, if sum ¡ 0 then it should be bounded by -1, this will produce a value for each perceptron, these results are then combined using:
$PerceptValue[i] * PerceptValue[i+1]. * PerceptValue[n]$
Each network then makes its output known, if both networks have the same output then they have synchronized for the current round and apply a synchronization rule where we alter the weights of each perceptron based on the anti-hebbian learning rule. This will result in one of two things to happen:

1. The difference in the weights from each network will stay the same in which point each node will move towards a boundary l.

2. The difference in weights will drop by one as one of the networks weight falls outside the boundary of l and is reset to l.

This will continue to happen until one network has decided that both networks are the same, this is done through a value based on the amount of consecutive times each network classify themselves as synchronized, if the synchronization value is too small then each network has a low chance of synchronizing, if it is too high then it has a high chance of synchronizing however, this may make it easier for attackers to synchronize with A and B if such attackers exist.

2 The problem

1. What N, K, L values allow for A and B to synchronize, are there inputs which perform better than others?

2. How do the values of N, K, L effect the chance of attackers succeeding in their attack?

3. How does the number of attackers effect the chance of succeeding in attack?

3 How I decided to combat the problem

Initially upon carrying out the KKK algorithm I found the base case of (1,1,1) did not synchronize unless each network was initially synchronized, this was because it oscillated between (0,1) and (1,0), because of this I decided it would be interesting to look into the values for N,K,L which successfully synchronized for A and B, I decided to set boundaries of this question to N¡=20, K¡=20 and L¡=10, to speed up the process of finding these values I removed the attacker as this would consume processing time, to further speed up this process I used openMP to test multiple systems of N,K,L at the same time.
Following this I decided to make a program which took these successful N, K, Ls, I decided to consider the max amount of synchronization needed to synchronize A and B successfully, as well as the max epochs needed. I did this by testing each combination of successful N, K, L serially before trying to attack these combinations of N, K, L.
Following this I ran the encryption algorithm with 1 attacker with the values found for max synchronization and epochs. I decided in the real world we would set our synchronization cap to the max synchronization steps required, as to reduce the chance of attackers succeeding in their attack.
From this I decided to consider how having multiple attackers with differing weights for their own neural network with the same network A and B effected the chance of attacker success, to do this efficiently I decided to use

omp and mpi.
Mpi was used to distribute the random seed that was used to create neural network A and B as well as the input values that were generated at random, this was to reduce mpi communication to the minimal amounts as each process would be able to work out values for A and B, which I considered less time intensive than transferring each input value. This was guaranteed to work if the machines that this was running on were using similar hardware as srand(seed) creates random numbers from a culmination of data about hardware of machines as well as the input supplied to srand().
Next, I decided to use omp to utilize multiple cores on each node in the cluster, these cores would each be assigned an individual neural network C, this allowed me to create i*j attackers, i being the number of cores that omp was using and j being the number of nodes.
For each combination of N,K,L I decided to run it through the system 100 times, making sure that neural networks A,B and C were different for each iteration this was to provide a more consistent set of results as if I were to test different attackers on the same A and B repeatedly then it is fair to presume a case that a network A and B are almost synchronized from the get go and therefore take less time to synchronize thus making it harder for the attacker to successfully break the encryption.
To ensure that I had adequate time to ensure results were obtained before deadline I decided to cap the epoch limit to 200,000 as well as having a synchronization limit of 2,000.

4 Results

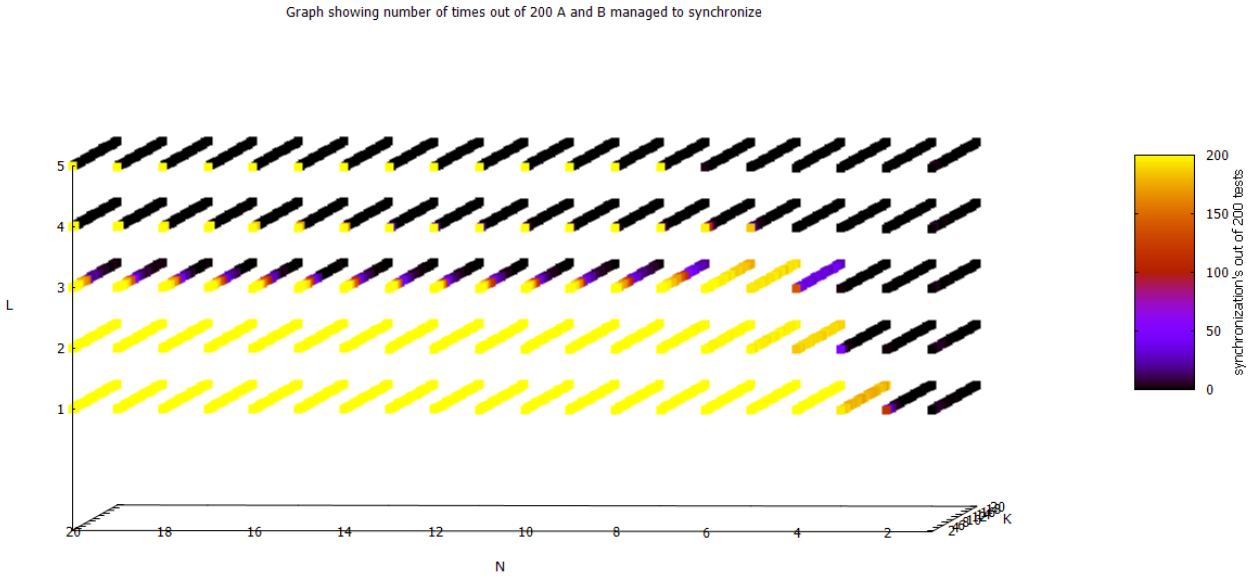

Figure 1: Chance of A and B Synchronizing L = [1:5] Appendix 1.3

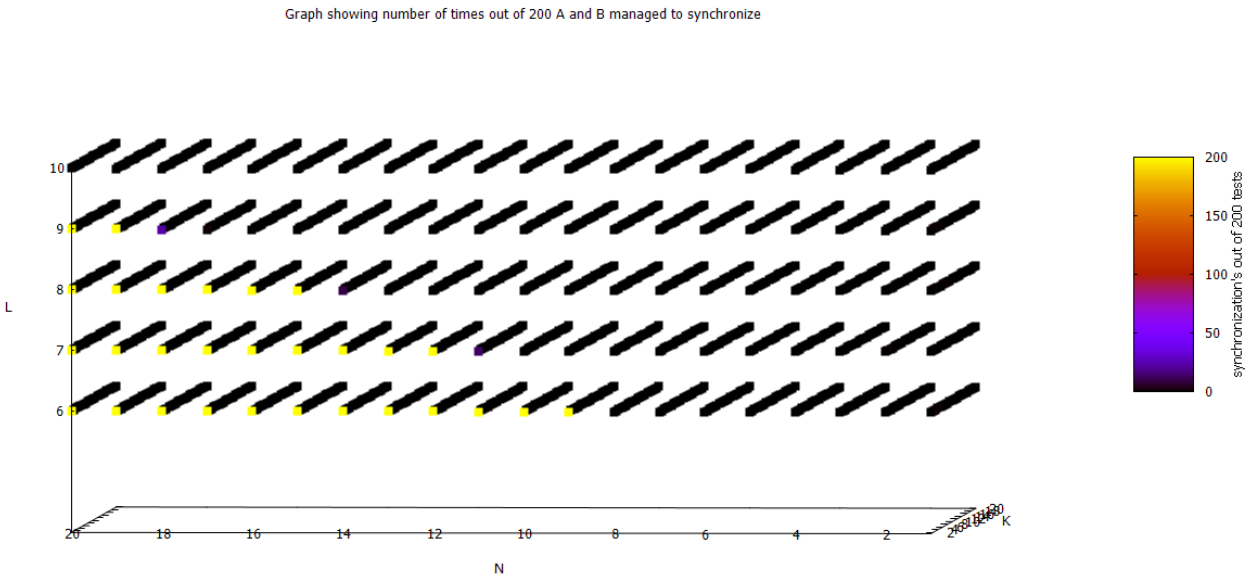

Figure 2: Chance of A and B Synchronizing L = [6:10] Appendix 1.3

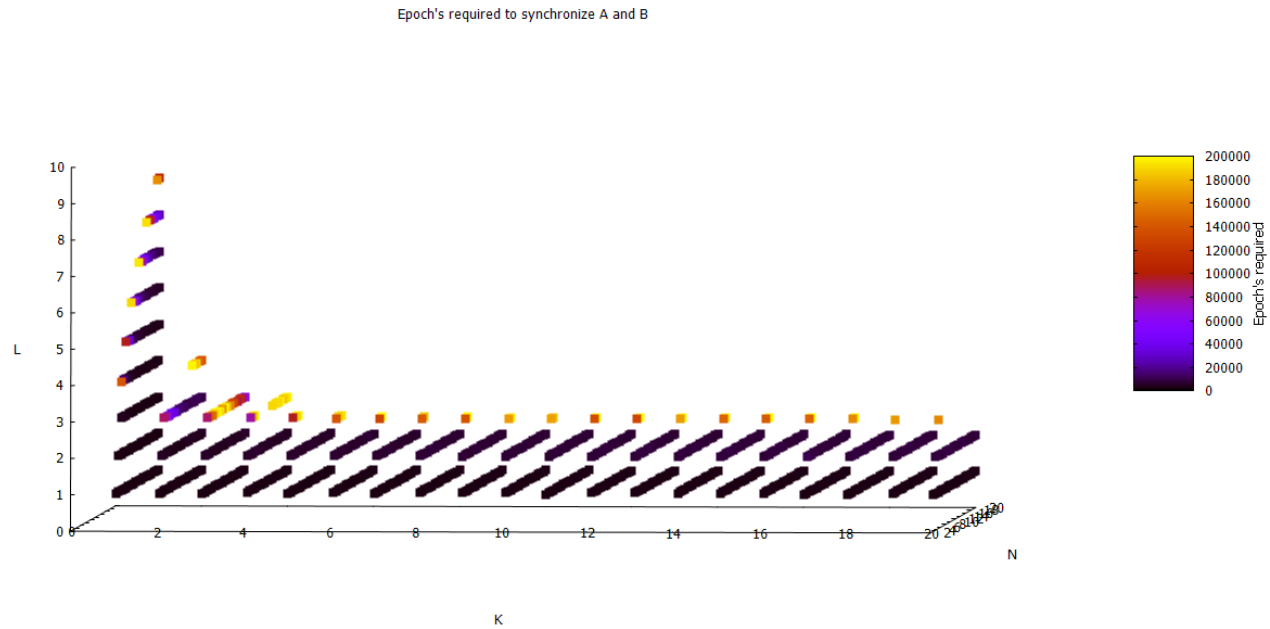

Figure 3: Epoch required for A and B to synchronize Appendix 2.3

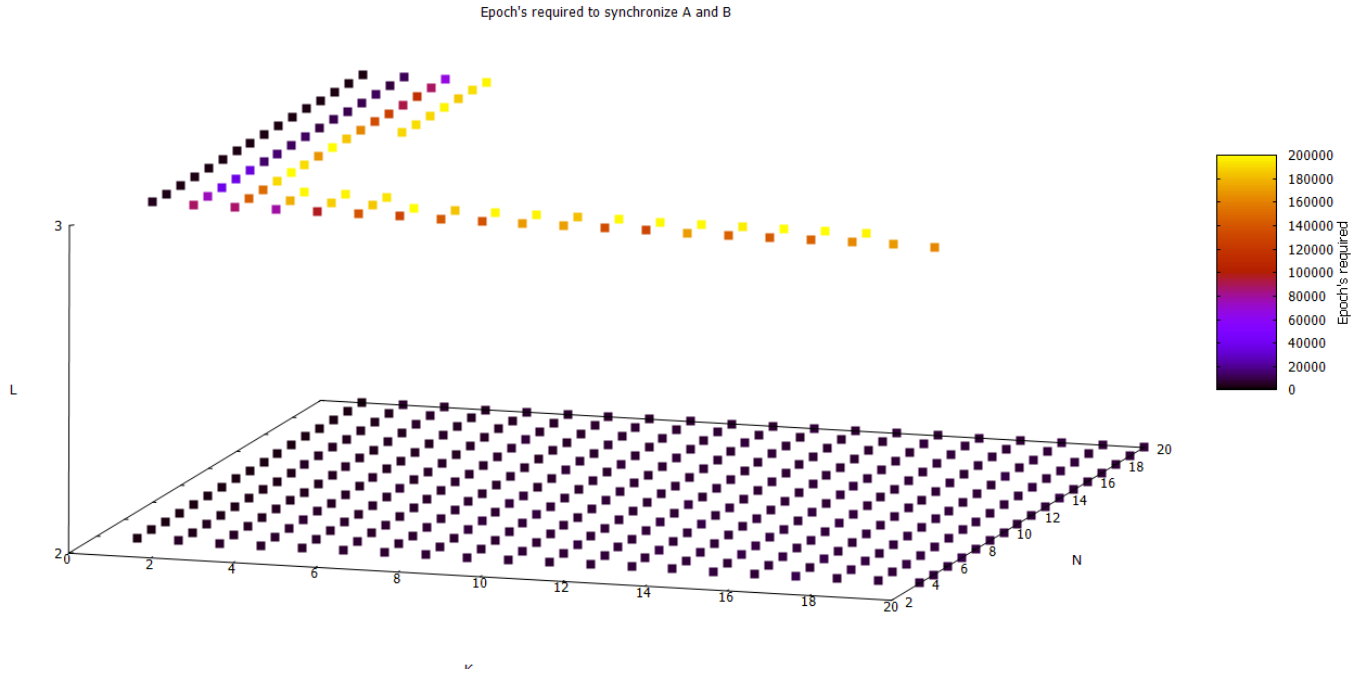

Figure 4: Epoch required for A and B to synchronize L = [2:3] Appendix 2.3

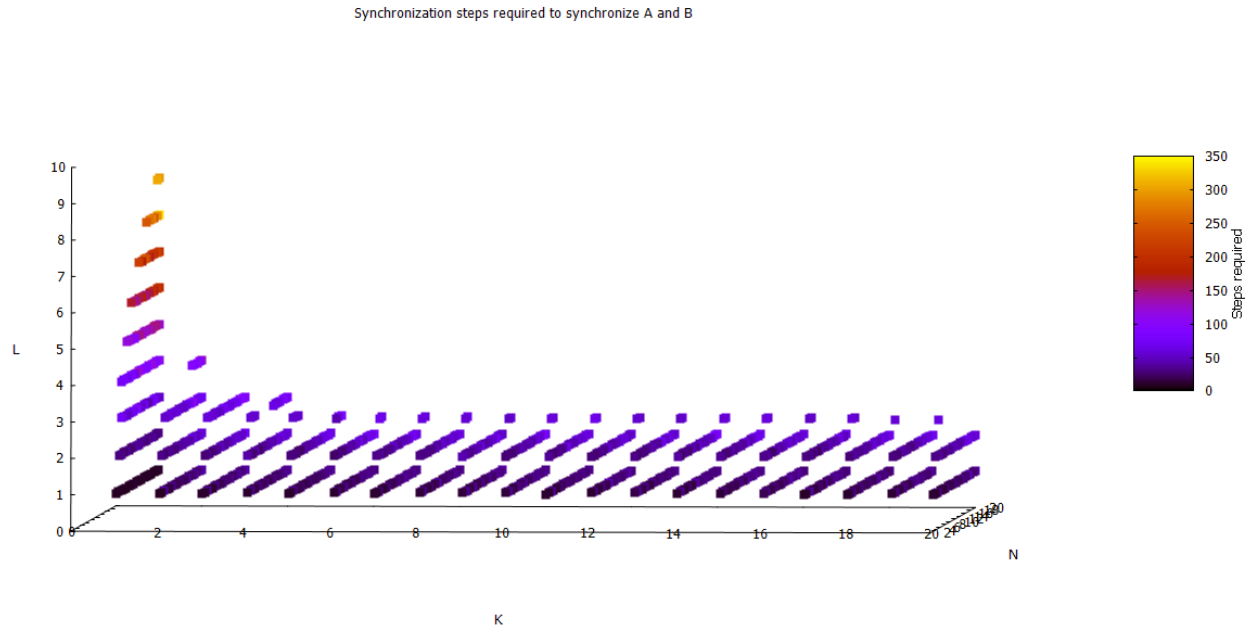

Figure 5: Synchronization steps to ensure synchronization Appendix 2.3

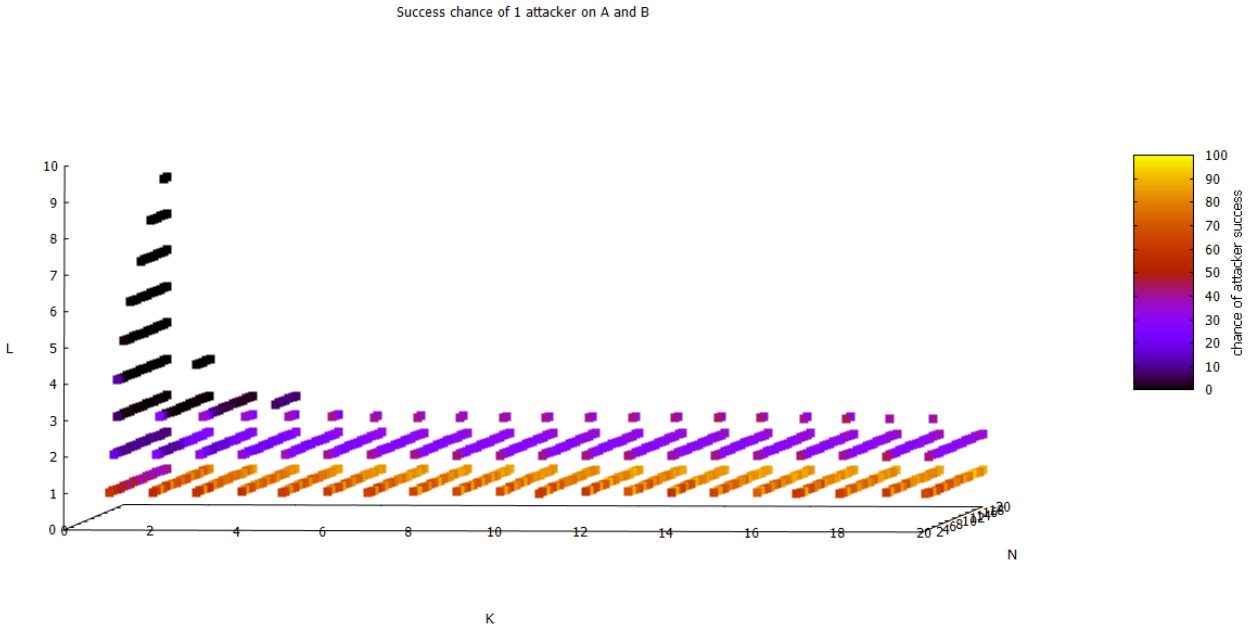

Figure 6: Chance of success attacking A and Bs communication with 1 attacker Appendix 2.3

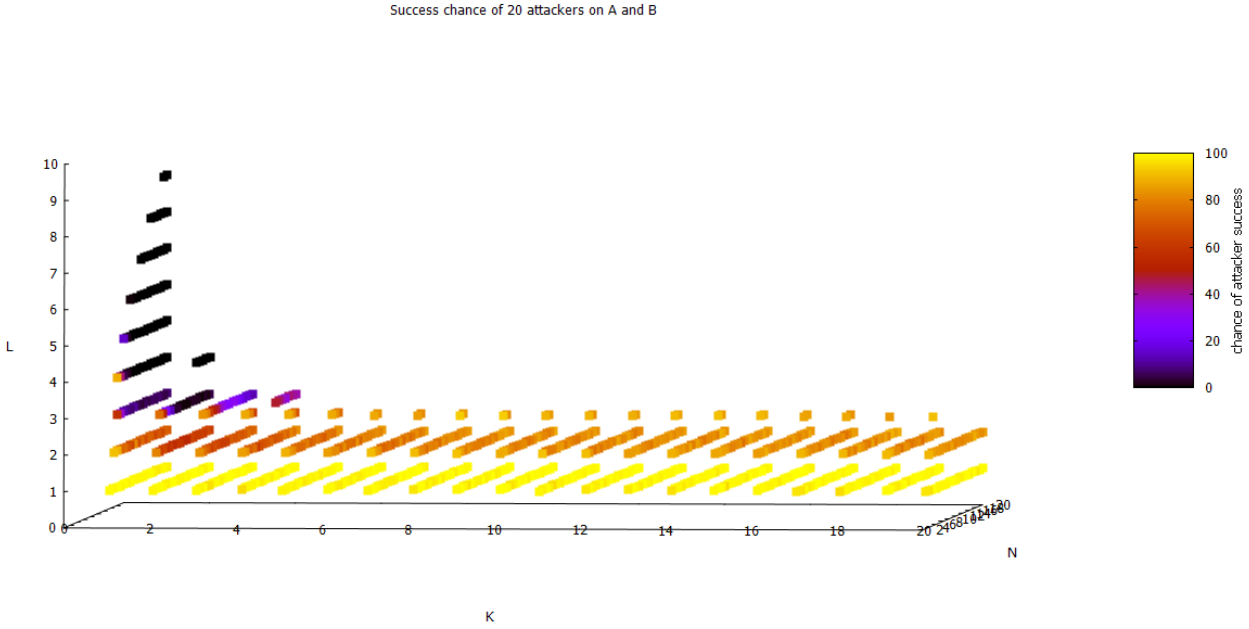

Figure 7: Chance of success attacking A and Bs communication with 20 attackers Appendix 2.4

5 Evaluation of Results

5.1 Effect of increases in N, K, L on A and Bs chance to synchronize.

Firstly, when looking at figure 1 and 2 we can see for N = 1 or 2 the chance of synchronization is extremely low, if they are not already synchronized to begin with, as stated earlier, when N is 1 it results in some oscillation around L = 0 I theorize the same may happen for N equals 2. From the figure we also see that increasing N also increases the chance for A and B to synchronize all the way to N = 20 however increasing K has the opposite effect reducing the chance of success for synchronization. Finally increasing L seems to decrease the chance of successful synchronization.
Because of this we can see that we should be able to increase our N without having to consider the chance of synchronization of A and B if K is relatively small. This opens a question what is restricting the chance of synchronization between A and B for higher Ks and Ls?
We can see from figure 3, 4 and 5 which show the amount of synchronization steps as well as epochs required for these specific combinations of N,K and L. We see that increasing our K value massively increases the amount of epochs required to synchronize A and B for example with L only being set to 3, we see that the required epochs is very close to our maximum epochs allowed (200,000) therefore K seems to be bounded by epoch limit this seems to be the case for L as well, as L increases so does the amount of epochs required however, the rate at which the epoch count required increases is not as drastic as that of increasing K.
Finally, as we can see the reason why it appears increasing N increases the chance of synchronization. It seems to reduce the number of epochs required to synchronize, however if we now look at figure 5, we can see that it also increases the amount of synchronization steps required, this increase is slow however we can assume that if we were to continue increasing N we would find a point where our model was restricted by our synchronization steps rather than epochs required.
From this we find our restrictions for each variable [N, K, L] are respectively [Synchronization limit, Epoch limit, Epoch limit].
Because N decreases the number of epochs required and L increases the number of epochs required we can consider a case at where we are restricted by the Synchronization limit as well as our epoch limit for our current model.

5.2 Effect of increases in N, K, L on 1 attacker

Looking at figure 6 we can see that increasing the value of K increases the chance of success for an attacker. However increasing N decreases this chance of success this is prominent as soon as L = 3.
Finally, we can see that increasing L has a drastic effect on the attackers chance, we can see that increasing from L=1 to L=2 resulted in a change of about 20% this becomes even more obvious at higher Ls. From this we can see that the best strategy for Users A and B is to increase N and L as much as possible which as we have discussed previously is possible up until our problem becomes bounded by both our synchronization limit and our epoch limit.

5.3 Effect of increases in N, K, L on 20 attackers

Looking at figure 6 and 7 we can see that increasing the number of attackers seems to increase the chance of a successful attack, however even with 20 attackers with only an L of 5 all attackers failed in attacking A and B. However, there was still an increase therefore it is possible with a high number of attackers we would be able to break all combinations of N, K and L this makes sense as each possible combination of N, K, L can only produce a limited amount of different neural networks. Therefore, increasing our number of attackers to N*K*(2L+1) would provide us with all possible neural networks that A and B could have, however this is very computationally difficult as with even N=1, K=10, L=3 we would have 30 possible combinations that our neural networks could take. Seeing as the values of K and L can be increased with relatively little chance in epoch count these values could turn out to be very large with very little computational work being required to synchronize them.

6 Appendix

6.1 Code used to determine N, K, L that synchronize

6.1.1 Main Program

```
#include "ModifiedHeader.h"
#include <omp.h>
#include <stdlib.h>
// set epoch limit and synchronization limit.
int EPOCH_LIMIT = 200000;
int SYNCHRONISATION_THRESHOLD = 2000;

int main(int argc, char *argv[])
{
  // Do each n in parallel
  #pragma omp parallel for num_threads(3) schedule(guided)
  for(int n = 1; n <= 20; n++)
  {
    for(int k = 1; k <= 20; k++)
    {
      for(int l = 1; l <= 10; l++)
      {
        // check how many iterations have been carried out
        int iterations = 0;
        // how many of this n,k,l combination have been correctly synchronized
        int testPositive = 0;
        do
        {
          // do 50 tests
          for(int j = 0; j < 50; j++)
          {
            // create a random seed for the specific test
            srand((int)time(NULL)+j+omp_get_thread_num());
            // construct a neural network A and B.
            struct NeuralNetwork neuralNetA = constructNeuralNetwork(n, k, l);
            struct NeuralNetwork neuralNetB = constructNeuralNetwork(n, k, l);
            // if the neural networks are synchronized to begin with then remake A and B.
            while(compareNetworks(neuralNetA, neuralNetB, n, k))
            {
              freeMemoryForNetwork(neuralNetA, n, k);
              freeMemoryForNetwork(neuralNetB, n, k);
              neuralNetA = constructNeuralNetwork(n, k, l);
              neuralNetB = constructNeuralNetwork(n, k, l);
            }
            // malloc space for inputs.
            int** inputs = malloc(sizeof (int*) * n);
            for (int i = 0; i < n; i++)
            {
              inputs[i] = malloc(sizeof (int) * k);
            }
            // get random inputs and assign it to inputs
            getRandomInputs(inputs, n, k);
            // perform KanterKinzelKanter Algorithm on neural network A and B.
            bool status = KanterKinzelKanter(neuralNetA, neuralNetB, inputs, n, k,
              l, SYNCHRONISATION_THRESHOLD, EPOCH_LIMIT);
            // free the inputs used.
            for (int i = 0; i < n; i++)
            {
              free(inputs[i]);
            }
            free(inputs);
            // if the networks were synchronized then add 1 to synchronization count.
            if(status == true && compareNetworks(neuralNetA, neuralNetB, n, k))
            {
              testPositive++;
            }
            // free memory for networks.
            freeMemoryForNetwork(neuralNetA, n, k);
            freeMemoryForNetwork(neuralNetB, n, k);
          }
          // one iteration complete
          iterations++;
          // continue testing unless we have reached 4 iterations (200 tests) or if we have
          // synchronized less than 50% of the time.
        }
        while(testPositive>(50*(double)iterations/2)&&iterations<4);
        // print out the results.
        printf("%d,%d,%d,%d\n",n,k,l,testPositive);
      }
    }
  }
  return 0;
}
```

6.1.2 Header file

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

// structure to create a neural network
struct NeuralNetwork
{
  int** weights;
  int* hiddenLayerOutputs;
  int networkOutput;
} neuralNet;

// create a definition for booleans
typedef enum
{
  true, false
} bool;

// pointers for methods.
bool KanterKinzelKanter(struct NeuralNetwork, struct NeuralNetwork, int**, int, int, int, int, int, int*, int*);
struct NeuralNetwork constructNeuralNetwork(int, int, int);
void initWeights(int**, int, int, int);
void updateWeights(int*, struct NeuralNetwork, int**, int, int, int);
int binaryRand(void);
int** getRandomInputs(int**, int, int);
int* getHiddenLayerOutputs(int*, struct NeuralNetwork, int**, int, int);
int getNetworkOutput(int*, struct NeuralNetwork, int**, int, int);
void freeMemoryForNetwork(struct NeuralNetwork, int, int);
char* printNetworkWeights(char*, struct NeuralNetwork, int, int, int, int);
bool compareNetworks(struct NeuralNetwork, struct NeuralNetwork ,int , int );
int numPlaces (int);

// method to check if two networks are identical. returns true if they arn't different else it returns false.
bool compareNetworks(struct NeuralNetwork neuralNetA, struct NeuralNetwork neuralNetB ,int k , int n)
{
  bool isDifferent = false;
  for(int i = 0 ; i < k; i++)
  {
    for(int j = 0; j < n; j++)
    {
      if(neuralNetA.weights[i][j]!=neuralNetB.weights[i][j])
      {
        isDifferent = true;
      }
    }
  }
  return isDifferent;
}

// KanterKinzelKanter algorithm takes a neural network a and a neural network b and
// attempts to synchronize them using the anti-hebbian rule.
bool KanterKinzelKanter(struct NeuralNetwork neuralNetA, struct NeuralNetwork neuralNetB, int** inputs, int k,
        int n, int l, int syncThreshold, int epochLimit)
{
// synchronization count.
  int s = 0;
  // epoch count.
  int epoch = 0;
  // malloc memory to create hidden layer outputs.
  int* hlOutputs = malloc(sizeof (int) * k);
  // whilst we havn't reached our max epoch's and max synchronization continue
  while ((s < syncThreshold) && (epoch < epochLimit))
  {
    get the outputs from each network using inputs.
    int outputA = getNetworkOutput(hlOutputs, neuralNetA, inputs, k, n);
    int outputB = getNetworkOutput(hlOutputs, neuralNetB, inputs, k, n);
    // if their outputs are equal update their weights using the anti-hebbian rule and add 1 to synchronization count
    if(outputA==outputB)
    {
      updateWeights(hlOutputs, neuralNetA, inputs, k, n, l);
      updateWeights(hlOutputs, neuralNetB, inputs, k, n, l);
      s=s+1;
    }
    else // else reset the synchronization cap.
    {
      s = 0;
    }
    getRandomInputs(inputs, k, n); // get new random inputs. and increase our epoch's
    epoch ++;
  }
  free(hlOutputs);
  if (s == syncThreshold)  // free our hidden layer ouputs.
  {
    return true;
  }
  return false;
}

// method to construct a neural network with k percepts, n weights and l input size and returns the network
struct NeuralNetwork constructNeuralNetwork(int k, int n, int l)
{
// create a neural network structure, malloc it and malloc space for each weight.
  struct NeuralNetwork neuralNetwork;
  neuralNetwork.weights = malloc(sizeof (int*) * (k));
  neuralNetwork.hiddenLayerOutputs = malloc(sizeof (int) * k);
  for (int i = 0; i < k; i++)
  {
    neuralNetwork.weights[i] = malloc(sizeof (int) * n);
    for (int j = 0; j < n; j++)
    {
      // set the weight of a specific percept to rand()
      neuralNetwork.weights[i][j] = rand() % (2 * l + 1) - l;
    }
  }
  return neuralNetwork;
}
```

```
// update the weights using anti-hebbian rule.
void updateWeights(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k, int n, int l)
{
  // get the hidden layer outputs.
  getHiddenLayerOutputs(hlOutputs, neuralNet, inputs, k, n);
  // for each percept
  for (int i = 0; i < k; i++)
  {
    // for each weight inside the percept
    for (int j = 0; j < n; j++)
    {
      // get its anti-hebbian value.
      neuralNet.weights[i][j] = neuralNet.weights[i][j] + (hlOutputs[i]*inputs[i][j]);
      // if it is less than 1 set it to -1 else set it to +1.
      if (neuralNet.weights[i][j] < ((-1) * 1))
      {
        neuralNet.weights[i][j] = (-1) * 1;
      }
      else if (neuralNet.weights[i][j] > 1)
      {
        neuralNet.weights[i][j] = 1;
      }
    }
  }
}

// create a random number that is either 1 or -1.
int binaryRand()
{
  int randNum = rand();
  if (randNum % 2 == 0)
  {
    return 1;
  }
  else
  {
    return -1;
  }
}

// create random inputs which are either -1 or 1.
int** getRandomInputs(int** inputs, int k, int n)
{
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < n; j++)
    {
      inputs[i][j] = binaryRand();
    }
  }
  return inputs;
}

// returns the hidden layer outputs.
int* getHiddenLayerOutputs(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k, int n)
{
  // for each percept
  for (int i = 0; i < k; i++)
  {
    // get the sum of each weight multiplied by each input value.
    int sum = 0;
    for (int j = 0; j < n; j++)
    {
        sum = sum - (neuralNet.weights[i][j] * inputs[i][j]);
    }
    // if the value of the sum is less than 0 then assign it -1 else assign it +1.
    if (sum <= 0)
    {
      hlOutputs[i] = -1;
    }
    else
    {
      hlOutputs[i] = 1;
    }
  }
  // return the hidden layer outputs.
  return hlOutputs;
}

// get the output from the percept.
int getNetworkOutput(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k, int n)
{
  // for each hidden layer output
  getHiddenLayerOutputs(hlOutputs, neuralNet, inputs, k, n);
  int prod = 1;
  // multiply each percepts hidden layer output to find out the global output for a network.
  for (int i = 0; i < k; i++)
  {
    prod = prod * (hlOutputs[i]);
  }
  return prod;
}

// free the memory needed to create a network.
void freeMemoryForNetwork(struct NeuralNetwork neuralNet, int k, int n)
{
  for (int i = 0; i < k; i++)
  {
    free(neuralNet.weights[i]);
  }
  free(neuralNet.weights);
  free(neuralNet.hiddenLayerOutputs);
}
```

6.1.3 Values returned by program in K, N, L and times out of 200 Synchronized.

```
1 1 1 0
2 1 1 6
3 1 1 5
4 1 1 4
5 1 1 3
6 1 1 0
7 1 1 1
8 1 1 0
9 1 1 0
10 1 1 1
11 1 1 0
12 1 1 0
13 1 1 0
14 1 1 0
15 1 1 0
16 1 1 0
17 1 1 0
18 1 1 0
19 1 1 0
20 1 1 0
1 2 1 116
2 2 1 31
3 2 1 17
4 2 1 4
5 2 1 3
6 2 1 0
7 2 1 0
8 2 1 0
9 2 1 0
10 2 1 0
11 2 1 0
12 2 1 0
13 2 1 0
14 2 1 0
15 2 1 0
16 2 1 0
17 2 1 0
18 2 1 0
19 2 1 0
20 2 1 0
1 3 1 195
2 3 1 185
3 3 1 189
4 3 1 175
5 3 1 181
6 3 1 176
7 3 1 180
8 3 1 179
9 3 1 170
10 3 1 175
11 3 1 183
12 3 1 177
13 3 1 174
14 3 1 174
15 3 1 176
16 3 1 179
17 3 1 184
18 3 1 181
19 3 1 172
20 3 1 186
1 4 1 200
2 4 1 200
3 4 1 200
4 4 1 200
5 4 1 200
6 4 1 200
7 4 1 200
8 4 1 200
9 4 1 200
10 4 1 200
11 4 1 200
12 4 1 200
13 4 1 200
14 4 1 200
15 4 1 200
16 4 1 200
17 4 1 200
18 4 1 200
19 4 1 200
20 4 1 200
1 5 1 200
2 5 1 200
3 5 1 200
4 5 1 200
5 5 1 200
6 5 1 200
7 5 1 200
8 5 1 200
9 5 1 200
10 5 1 200
11 5 1 200
12 5 1 200
13 5 1 200
14 5 1 200
15 5 1 200
16 5 1 200
17 5 1 200
18 5 1 200
19 5 1 200
20 5 1 200
1 6 1 200
2 6 1 200
3 6 1 200
4 6 1 200
5 6 1 200
6 6 1 200
7 6 1 200
8 6 1 200
9 6 1 200
10 6 1 200
11 6 1 200
12 6 1 200
13 6 1 200
14 6 1 200
15 6 1 200
16 6 1 200
17 6 1 200
18 6 1 200
19 6 1 200
20 6 1 200
1 7 1 200
2 7 1 200
3 7 1 200
4 7 1 200
5 7 1 200
6 7 1 200
7 7 1 200
8 7 1 200
9 7 1 200
10 7 1 200
11 7 1 200
12 7 1 200
13 7 1 200
14 7 1 200
15 7 1 200
16 7 1 200
17 7 1 200
18 7 1 200
19 7 1 200
20 7 1 200
1 8 1 200
2 8 1 200
3 8 1 200
4 8 1 200
5 8 1 200
6 8 1 200
7 8 1 200
8 8 1 200
9 8 1 200
10 8 1 200
11 8 1 200
12 8 1 200
13 8 1 200
14 8 1 200
15 8 1 200
16 8 1 200
17 8 1 200
18 8 1 200
19 8 1 200
20 8 1 200
1 9 1 200
2 9 1 200
3 9 1 200
4 9 1 200
5 9 1 200
6 9 1 200
7 9 1 200
8 9 1 200
9 9 1 200
10 9 1 200
11 9 1 200
12 9 1 200
13 9 1 200
14 9 1 200
15 9 1 200
16 9 1 200
17 9 1 200
18 9 1 200
19 9 1 200
20 9 1 200
1 10 1 200
2 10 1 200
3 10 1 200
4 10 1 200
5 10 1 200
6 10 1 200
7 10 1 200
8 10 1 200
9 10 1 200
10 10 1 200
11 10 1 200
12 10 1 200
13 10 1 200
14 10 1 200
15 10 1 200
16 10 1 200
17 10 1 200
18 10 1 200
19 10 1 200
20 10 1 200
1 11 1 200
2 11 1 200
3 11 1 200
4 11 1 200
5 11 1 200
6 11 1 200
7 11 1 200
8 11 1 200
9 11 1 200
10 11 1 200
11 11 1 200
12 11 1 200
13 11 1 200
14 11 1 200
15 11 1 200
16 11 1 200
17 11 1 200
18 11 1 200
19 11 1 200
20 11 1 200
1 12 1 200
2 12 1 200
3 12 1 200
4 12 1 200
5 12 1 200
6 12 1 200
7 12 1 200
8 12 1 200
9 12 1 200
10 12 1 200
11 12 1 200
12 12 1 200
13 12 1 200
14 12 1 200
15 12 1 200
16 12 1 200
17 12 1 200
18 12 1 200
19 12 1 200
20 12 1 200
1 13 1 200
2 13 1 200
3 13 1 200
4 13 1 200
5 13 1 200
6 13 1 200
7 13 1 200
8 13 1 200
9 13 1 200
10 13 1 200
11 13 1 200
12 13 1 200
13 13 1 200
14 13 1 200
15 13 1 200
16 13 1 200
17 13 1 200
18 13 1 200
19 13 1 200
20 13 1 200
1 14 1 200
2 14 1 200
3 14 1 200
4 14 1 200
5 14 1 200
6 14 1 200
7 14 1 200
8 14 1 200
9 14 1 200
10 14 1 200
11 14 1 200
12 14 1 200
13 14 1 200
14 14 1 200
15 14 1 200
16 14 1 200
17 14 1 200
18 14 1 200
19 14 1 200
20 14 1 200
1 15 1 200
2 15 1 200
3 15 1 200
4 15 1 200
5 15 1 200
6 15 1 200
7 15 1 200
8 15 1 200
9 15 1 200
10 15 1 200
11 15 1 200
12 15 1 200
13 15 1 200
14 15 1 200
15 15 1 200
16 15 1 200
17 15 1 200
18 15 1 200
19 15 1 200
20 15 1 200
1 16 1 200
2 16 1 200
3 16 1 200
4 16 1 200
5 16 1 200
6 16 1 200
7 16 1 200
8 16 1 200
9 16 1 200
10 16 1 200
11 16 1 200
12 16 1 200
13 16 1 200
14 16 1 200
15 16 1 200
16 16 1 200
17 16 1 200
18 16 1 200
19 16 1 200
20 16 1 200
1 17 1 200
2 17 1 200
3 17 1 200
4 17 1 200
5 17 1 200
6 17 1 200
7 17 1 200
8 17 1 200
9 17 1 200
10 17 1 200
11 17 1 200
12 17 1 200
13 17 1 200
14 17 1 200
15 17 1 200
16 17 1 200
17 17 1 200
18 17 1 200
19 17 1 200
20 17 1 200
1 18 1 200
2 18 1 200
3 18 1 200
4 18 1 200
5 18 1 200
6 18 1 200
7 18 1 200
8 18 1 200
9 18 1 200
10 18 1 200
11 18 1 200
12 18 1 200
13 18 1 200
14 18 1 200
15 18 1 200
16 18 1 200
17 18 1 200
18 18 1 200
19 18 1 200
20 18 1 200
1 19 1 200
2 19 1 200
3 19 1 200
4 19 1 200
5 19 1 200
6 19 1 200
7 19 1 200
8 19 1 200
9 19 1 200
10 19 1 200
11 19 1 200
12 19 1 200
13 19 1 200
14 19 1 200
15 19 1 200
16 19 1 200
17 19 1 200
18 19 1 200
19 19 1 200
20 19 1 200
1 20 1 200
2 20 1 200
3 20 1 200
4 20 1 200
5 20 1 200
6 20 1 200
7 20 1 200
8 20 1 200
9 20 1 200
10 20 1 200
11 20 1 200
12 20 1 200
13 20 1 200
14 20 1 200
15 20 1 200
16 20 1 200
17 20 1 200
18 20 1 200
19 20 1 200
20 20 1 200
1 1 2 0
2 1 2 4
3 1 2 5
4 1 2 5
5 1 2 3
6 1 2 0
7 1 2 0
8 1 2 0
9 1 2 0
10 1 2 0
11 1 2 0
12 1 2 0
13 1 2 0
14 1 2 0
15 1 2 0
16 1 2 0
17 1 2 0
18 1 2 0
19 1 2 0
20 1 2 0
1 2 2 1
2 2 2 1
3 2 2 0
4 2 2 0
5 2 2 0
6 2 2 0
7 2 2 0
8 2 2 0
9 2 2 0
10 2 2 0
11 2 2 0
12 2 2 0
13 2 2 0
14 2 2 0
15 2 2 0
16 2 2 0
17 2 2 0
18 2 2 0
19 2 2 0
20 2 2 0
1 3 2 47
2 3 2 8
3 3 2 1
4 3 2 1
5 3 2 0
6 3 2 0
7 3 2 0
8 3 2 0
9 3 2 0
10 3 2 0
11 3 2 0
12 3 2 0
13 3 2 0
14 3 2 0
15 3 2 0
16 3 2 0
17 3 2 0
18 3 2 0
19 3 2 0
20 3 2 0
1 4 2 184
2 4 2 189
3 4 2 185
4 4 2 191
5 4 2 189
6 4 2 192
7 4 2 188
8 4 2 191
9 4 2 188
10 4 2 191
11 4 2 187
12 4 2 185
13 4 2 193
14 4 2 189
15 4 2 189
16 4 2 190
17 4 2 191
18 4 2 188
19 4 2 189
20 4 2 187
1 5 2 199
2 5 2 193
3 5 2 199
4 5 2 196
5 5 2 195
6 5 2 196
7 5 2 195
8 5 2 193
9 5 2 198
10 5 2 196
11 5 2 195
12 5 2 197
13 5 2 200
14 5 2 195
15 5 2 197
16 5 2 196
17 5 2 197
18 5 2 196
19 5 2 199
20 5 2 198
1 6 2 200
2 6 2 198
3 6 2 200
4 6 2 197
5 6 2 199
6 6 2 199
7 6 2 199
8 6 2 198
9 6 2 200
10 6 2 198
11 6 2 197
12 6 2 199
13 6 2 199
14 6 2 200
15 6 2 199
16 6 2 199
17 6 2 199
18 6 2 200
19 6 2 200
20 6 2 199
1 7 2 200
2 7 2 200
3 7 2 200
4 7 2 200
5 7 2 200
6 7 2 200
7 7 2 199
8 7 2 200
9 7 2 199
10 7 2 200
11 7 2 200
12 7 2 200
13 7 2 200
14 7 2 200
15 7 2 199
16 7 2 198
17 7 2 200
18 7 2 199
19 7 2 200
20 7 2 198
1 8 2 200
2 8 2 200
3 8 2 200
4 8 2 200
5 8 2 200
6 8 2 200
7 8 2 200
8 8 2 200
9 8 2 200
10 8 2 200
11 8 2 200
12 8 2 200
13 8 2 200
14 8 2 200
15 8 2 200
16 8 2 200
17 8 2 200
18 8 2 200
19 8 2 200
20 8 2 200
1 9 2 200
2 9 2 200
3 9 2 200
4 9 2 200
5 9 2 200
6 9 2 200
7 9 2 200
8 9 2 200
9 9 2 200
10 9 2 200
11 9 2 200
12 9 2 200
13 9 2 200
14 9 2 200
15 9 2 200
16 9 2 200
17 9 2 200
18 9 2 200
19 9 2 200
20 9 2 200
1 10 2 200
2 10 2 200
3 10 2 200
4 10 2 200
5 10 2 200
6 10 2 200
7 10 2 200
8 10 2 200
9 10 2 200
10 10 2 200
11 10 2 200
12 10 2 200
13 10 2 200
14 10 2 199
15 10 2 200
16 10 2 200
17 10 2 200
18 10 2 200
19 10 2 200
20 10 2 200
1 11 2 200
2 11 2 200
3 11 2 200
4 11 2 200
5 11 2 200
6 11 2 200
7 11 2 200
8 11 2 200
9 11 2 200
10 11 2 200
11 11 2 200
12 11 2 200
13 11 2 200
14 11 2 200
15 11 2 200
16 11 2 200
17 11 2 200
18 11 2 200
19 11 2 200
20 11 2 200
1 12 2 200
2 12 2 200
3 12 2 200
4 12 2 200
5 12 2 200
6 12 2 200
7 12 2 200
8 12 2 200
9 12 2 200
10 12 2 200
11 12 2 200
12 12 2 200
13 12 2 200
14 12 2 200
15 12 2 200
16 12 2 200
17 12 2 200
18 12 2 200
19 12 2 200
20 12 2 200
1 13 2 200
2 13 2 200
3 13 2 200
4 13 2 200
5 13 2 200
6 13 2 200
7 13 2 200
8 13 2 200
9 13 2 200
10 13 2 200
11 13 2 200
12 13 2 200
13 13 2 200
14 13 2 200
15 13 2 200
16 13 2 200
17 13 2 200
18 13 2 200
19 13 2 200
20 13 2 200
1 14 2 200
2 14 2 200
3 14 2 200
4 14 2 200
5 14 2 200
6 14 2 200
7 14 2 200
8 14 2 200
9 14 2 200
10 14 2 200
11 14 2 200
12 14 2 200
13 14 2 200
14 14 2 200
15 14 2 200
16 14 2 200
17 14 2 200
18 14 2 200
19 14 2 200
20 14 2 200
1 15 2 200
2 15 2 200
3 15 2 200
4 15 2 200
5 15 2 200
6 15 2 200
7 15 2 200
8 15 2 200
9 15 2 200
10 15 2 200
11 15 2 200
12 15 2 200
13 15 2 200
14 15 2 200
15 15 2 200
16 15 2 200
17 15 2 200
18 15 2 200
19 15 2 200
20 15 2 200
1 16 2 200
2 16 2 200
3 16 2 200
4 16 2 200
5 16 2 200
6 16 2 200
7 16 2 200
8 16 2 200
9 16 2 200
10 16 2 200
11 16 2 200
12 16 2 200
13 16 2 200
14 16 2 200
15 16 2 200
16 16 2 200
17 16 2 200
18 16 2 200
19 16 2 200
20 16 2 200
1 17 2 200
2 17 2 200
3 17 2 200
4 17 2 200
5 17 2 200
6 17 2 200
7 17 2 200
8 17 2 200
9 17 2 200
10 17 2 200
11 17 2 200
12 17 2 200
13 17 2 200
14 17 2 200
15 17 2 200
16 17 2 200
17 17 2 200
18 17 2 200
19 17 2 200
20 17 2 200
1 18 2 200
2 18 2 200
3 18 2 200
4 18 2 200
5 18 2 200
6 18 2 200
7 18 2 200
8 18 2 200
9 18 2 200
10 18 2 200
11 18 2 200
12 18 2 200
13 18 2 200
14 18 2 200
15 18 2 200
16 18 2 200
17 18 2 200
18 18 2 200
19 18 2 200
20 18 2 200
1 19 2 200
2 19 2 200
3 19 2 200
4 19 2 200
5 19 2 200
6 19 2 200
7 19 2 200
8 19 2 200
9 19 2 200
10 19 2 200
11 19 2 200
12 19 2 200
13 19 2 200
14 19 2 200
15 19 2 200
16 19 2 200
17 19 2 200
18 19 2 200
19 19 2 200
20 19 2 200
1 20 2 200
2 20 2 200
3 20 2 200
4 20 2 200
5 20 2 200
6 20 2 200
7 20 2 200
8 20 2 200
9 20 2 200
10 20 2 200
11 20 2 200
12 20 2 200
13 20 2 200
14 20 2 200
15 20 2 200
16 20 2 200
17 20 2 200
18 20 2 200
19 20 2 200
20 20 2 200
1 1 3 0
2 1 3 4
3 1 3 2
4 1 3 0
5 1 3 0
6 1 3 0
7 1 3 1
8 1 3 0
9 1 3 0
10 1 3 0
11 1 3 0
12 1 3 0
13 1 3 0
14 1 3 0
15 1 3 0
16 1 3 0
17 1 3 0
18 1 3 0
19 1 3 0
20 1 3 0
1 2 3 0
2 2 3 1
3 2 3 0
4 2 3 0
5 2 3 0
6 2 3 0
7 2 3 0
8 2 3 0
9 2 3 0
10 2 3 0
11 2 3 0
12 2 3 0
13 2 3 0
14 2 3 0
15 2 3 0
16 2 3 0
17 2 3 0
18 2 3 0
19 2 3 0
20 2 3 0
1 3 3 1
2 3 3 0
3 3 3 0
4 3 3 0
5 3 3 0
6 3 3 0
7 3 3 0
8 3 3 0
```

9 3 3 0
10 3 3 0
11 3 3 0
12 3 3 0
13 3 3 0
14 3 3 0
15 3 3 0
16 3 3 0
17 3 3 0
18 3 3 0
19 3 3 0
20 3 3 0
1 4 3 138
2 4 3 42
3 4 3 38
4 4 3 30
5 4 3 36
6 4 3 36
7 4 3 34
8 4 3 29
9 4 3 40
10 4 3 34
11 4 3 32
12 4 3 35
13 4 3 32
14 4 3 30
15 4 3 27
16 4 3 40
17 4 3 40
18 4 3 33
19 4 3 27
20 4 3 22
1 5 3 194
2 5 3 194
3 5 3 193
4 5 3 194
5 5 3 196
6 5 3 191
7 5 3 191
8 5 3 194
9 5 3 192
10 5 3 196
11 5 3 196
12 5 3 188
13 5 3 193
14 5 3 195
15 5 3 185
16 5 3 195
17 5 3 192
18 5 3 192
19 5 3 195
20 5 3 190
1 6 3 200
2 6 3 199
3 6 3 200
4 6 3 200
5 6 3 199
6 6 3 195
7 6 3 196
8 6 3 197
9 6 3 196
10 6 3 190
11 6 3 197
12 6 3 193
13 6 3 185
14 6 3 189
15 6 3 187
16 6 3 182
17 6 3 185
18 6 3 180
19 6 3 173
20 6 3 176
10 7 3 134
11 7 3 111
12 7 3 50
13 7 3 50
14 7 3 33
15 7 3 37
16 7 3 32
17 7 3 24
18 7 3 25
19 7 3 19
20 7 3 21
10 8 3 29
11 8 3 18
12 8 3 16
13 8 3 16
14 8 3 10
15 8 3 9
16 8 3 4
17 8 3 7
18 8 3 3
19 8 3 0
20 8 3 1
10 9 3 22
11 9 3 10
12 9 3 8
13 9 3 4
14 9 3 5
15 9 3 2
16 9 3 2
17 9 3 2
18 9 3 0
19 9 3 0
20 9 3 0
10 10 3 10
11 10 3 5
12 10 3 3
13 10 3 4
14 10 3 1
15 10 3 0
16 10 3 0
17 10 3 0
18 10 3 0
19 10 3 0
20 10 3 0
10 11 3 16
11 11 3 4
12 11 3 1
13 11 3 3
14 11 3 0
15 11 3 0
16 11 3 0
17 11 3 0
18 11 3 0
19 11 3 0
20 11 3 0
11 12 3 3
12 12 3 4
13 12 3 0
14 12 3 1
15 12 3 0
16 12 3 0
17 12 3 0
18 12 3 0
19 12 3 0
20 12 3 0
11 13 3 5
12 13 3 5
13 13 3 1
14 13 3 0
15 13 3 2
16 13 3 0
17 13 3 1
17 13 3 0
18 13 3 0
19 13 3 0
20 13 3 0
18 14 3 0
19 14 3 0
20 14 3 0
18 15 3 2
19 15 3 0
20 15 3 0
18 16 3 0
19 16 3 0
20 16 3 0
18 17 3 0
19 17 3 0
20 17 3 0
1 7 3 200
2 7 3 200
3 7 3 197
4 7 3 196
5 7 3 183
6 7 3 182
7 7 3 168
8 7 3 170
9 7 3 147
1 9 3 200
2 9 3 200
3 9 3 192
4 9 3 172
5 9 3 154
6 9 3 104
7 9 3 43
8 9 3 32
9 9 3 21
1 10 3 200
2 10 3 200
3 10 3 193
4 10 3 173
5 10 3 132
6 10 3 104
7 10 3 36
8 10 3 23
9 10 3 15
1 11 3 200
2 11 3 200
3 11 3 192
4 11 3 162
5 11 3 126
6 11 3 107
7 11 3 39
8 11 3 23
9 11 3 18
1 12 3 200
2 12 3 200
3 12 3 198
4 12 3 165
5 12 3 119
6 12 3 50
7 12 3 31
8 12 3 18
9 12 3 11
10 12 3 9
1 13 3 200
2 13 3 200
3 13 3 198
4 13 3 172
5 13 3 137
6 13 3 41
7 13 3 37
8 13 3 20
9 13 3 19
10 13 3 11
1 14 3 200
2 14 3 200
3 14 3 200
4 14 3 183
5 14 3 143
6 14 3 108
7 14 3 39
8 14 3 29
9 14 3 21
10 14 3 13
11 14 3 9
12 14 3 7
13 14 3 0
14 14 3 0
15 14 3 2
16 14 3 0
17 14 3 0
1 15 3 200
2 15 3 200
3 15 3 199
4 15 3 191
5 15 3 152
6 15 3 112
7 15 3 40
8 15 3 32
9 15 3 22
10 15 3 0
11 15 3 0
12 15 3 0
13 15 3 0
14 15 3 0
15 15 3 0
16 15 3 0
17 15 3 0
1 16 3 200
2 16 3 200
3 16 3 200
4 16 3 188
5 16 3 156
6 16 3 130
7 16 3 100
8 16 3 28
9 16 3 28
10 16 3 20
11 16 3 10
12 16 3 0
13 16 3 4
14 16 3 3
15 16 3 0
16 16 3 0
17 16 3 0
1 17 3 200
2 17 3 200
3 17 3 200
4 17 3 192
5 17 3 172
6 17 3 136
7 17 3 105
8 17 3 37
9 17 3 29
10 17 3 21
11 17 3 16
12 17 3 9
13 17 3 11
14 17 3 0
15 17 3 0
16 17 3 0
17 17 3 0
1 18 3 200
2 18 3 200
3 18 3 200
4 18 3 194
5 18 3 170
6 18 3 159
7 18 3 112
8 18 3 39
9 18 3 41
10 18 3 18
11 18 3 18
12 18 3 0
13 18 3 7
14 18 3 0
15 18 3 0
16 18 3 0
17 18 3 0
18 18 3 0
19 18 3 2
20 18 3 0
1 19 3 200
2 19 3 200
3 19 3 200
4 19 3 195
5 19 3 179
6 19 3 145
7 19 3 136
8 19 3 47
9 19 3 25
10 19 3 15
11 19 3 12
12 19 3 5
13 19 3 9
14 19 3 5
15 19 3 1
16 19 3 1
17 19 3 2
18 19 3 2
19 19 3 2
20 19 3 1
1 20 3 200
2 20 3 200
3 20 3 200
4 20 3 196
5 20 3 176
6 20 3 165
7 20 3 147
8 20 3 24
9 20 3 17
10 20 3 23
11 20 3 16
12 20 3 11
13 20 3 4
14 20 3 6
15 20 3 4
16 20 3 0
17 20 3 1
18 20 3 3
19 20 3 0
20 20 3 1
1 1 4 0
2 1 4 6
3 1 4 0
4 1 4 0
5 1 4 1
6 1 4 1
7 1 4 0
8 1 4 0
9 1 4 0
10 1 4 0
11 1 4 0
12 1 4 0
13 1 4 0
14 1 4 0
15 1 4 0
16 1 4 0
17 1 4 0
18 1 4 0
19 1 4 0
20 1 4 0
1 2 4 0
2 2 4 0
3 2 4 0
4 2 4 0
5 2 4 0
6 2 4 0
7 2 4 0
8 2 4 0
9 2 4 0
10 2 4 0
11 2 4 0
12 2 4 0
13 2 4 0
14 2 4 0
15 2 4 0
16 2 4 0
17 2 4 0
18 2 4 0
19 2 4 0
20 2 4 0
1 3 4 0
2 3 4 0
3 3 4 0
4 3 4 0
5 3 4 0
6 3 4 0
7 3 4 0
8 3 4 0
9 3 4 0
10 3 4 0
11 3 4 0
12 3 4 0
13 3 4 0
14 3 4 0
15 3 4 0
16 3 4 0
17 3 4 0
18 3 4 0
19 3 4 0
20 3 4 0
1 4 4 0
2 4 4 0
3 4 4 0
4 4 4 0
5 4 4 0
6 4 4 0
7 4 4 0
8 4 4 0
9 4 4 0
10 4 4 0
11 4 4 0
12 4 4 0
13 4 4 0
14 4 4 0
15 4 4 0
16 4 4 0
17 4 4 0
18 4 4 0
19 4 4 0
20 4 4 0
1 5 4 184
2 5 4 17
3 5 4 5
4 5 4 2
5 5 4 1
6 5 4 0
7 5 4 0
8 5 4 0
9 5 4 0
10 5 4 0
11 5 4 0
12 5 4 0
13 5 4 0
14 5 4 0
15 5 4 0
16 5 4 0
17 5 4 0
18 5 4 0
19 5 4 0
20 5 4 0
1 6 4 196
2 6 4 106
3 6 4 10
4 6 4 2
5 6 4 0
6 6 4 0
7 6 4 0
8 6 4 0
9 6 4 0
10 6 4 0
11 6 4 0
12 6 4 0
13 6 4 0
14 6 4 0
15 6 4 0
16 6 4 0
17 6 4 0
18 6 4 0
19 6 4 0
20 6 4 0
1 7 4 200
2 7 4 12
3 7 4 1
4 7 4 0
5 7 4 0
6 7 4 0
7 7 4 0
8 7 4 0
9 7 4 0
10 7 4 0
11 7 4 0
12 7 4 0
13 7 4 0
14 7 4 0
15 7 4 0
16 7 4 0
17 7 4 0
18 7 4 0
19 7 4 0
20 7 4 0
1 8 4 200
2 8 4 4
3 8 4 0
4 8 4 0
5 8 4 0
6 8 4 0
7 8 4 0
8 8 4 0
9 8 4 0
10 8 4 0
11 8 4 0
12 8 4 0
13 8 4 0
14 8 4 0
15 8 4 0
16 8 4 0
17 8 4 0
18 8 4 0
19 8 4 0
20 8 4 0
1 9 4 200
2 9 4 7
3 9 4 0
4 9 4 0
5 9 4 0
6 9 4 0
7 9 4 0
8 9 4 0
9 9 4 0
10 9 4 0
11 9 4 0
12 9 4 0
13 9 4 0
14 9 4 0
15 9 4 0
16 9 4 0
17 9 4 0
18 9 4 0
19 9 4 0
20 9 4 0
1 10 4 200
2 10 4 10
3 10 4 0
4 10 4 0
5 10 4 0
6 10 4 0
7 10 4 0
8 10 4 0
9 10 4 0
10 10 4 0
11 10 4 0
12 10 4 0
13 10 4 0
14 10 4 0
15 10 4 0
16 10 4 0
17 10 4 0
18 10 4 0
19 10 4 0
20 10 4 0
1 11 4 200
2 11 4 18
3 11 4 0
4 11 4 0
5 11 4 0
6 11 4 0
7 11 4 0
8 11 4 0
9 11 4 0
10 11 4 0
11 11 4 0
12 11 4 0
13 11 4 0
14 11 4 0
15 11 4 0
16 11 4 0
17 11 4 0
18 11 4 0
19 11 4 0
20 11 4 0
1 12 4 200
2 12 4 17
3 12 4 0
4 12 4 0
5 12 4 0
6 12 4 0
7 12 4 0
8 12 4 0
9 12 4 0
10 12 4 0
11 12 4 0
12 12 4 0
13 12 4 0
14 12 4 0
15 12 4 0
16 12 4 0
17 12 4 0
18 12 4 0
19 12 4 0
20 12 4 0
1 13 4 200
2 13 4 23
3 13 4 0
4 13 4 0
5 13 4 0
6 13 4 0
7 13 4 0
8 13 4 0
9 13 4 0
10 13 4 0
11 13 4 0
12 13 4 0
13 13 4 0
14 13 4 0
15 13 4 0
16 13 4 0
17 13 4 0
18 13 4 0
19 13 4 0
20 13 4 0
1 14 4 200
2 14 4 160
3 14 4 0
4 14 4 0
5 14 4 0
6 14 4 0
7 14 4 0
8 14 4 0
9 14 4 0
10 14 4 0
11 14 4 0
12 14 4 0
13 14 4 0
14 14 4 0
15 14 4 0
16 14 4 0
17 14 4 0
18 14 4 0
19 14 4 0
20 14 4 0
1 15 4 200
2 15 4 169
3 15 4 0
4 15 4 0
5 15 4 0
6 15 4 0
7 15 4 0
8 15 4 0
9 15 4 0
10 15 4 0
11 15 4 0
12 15 4 0
13 15 4 0
14 15 4 0
15 15 4 0
16 15 4 0
17 15 4 0
18 15 4 0
19 15 4 0
20 15 4 0
1 16 4 200
2 16 4 185
3 16 4 0
4 16 4 0
5 16 4 0
6 16 4 0
7 16 4 0
8 16 4 0
9 16 4 0
10 16 4 0
11 16 4 0
12 16 4 0
13 16 4 0
14 16 4 0
15 16 4 0
16 16 4 0
17 16 4 0
18 16 4 0
19 16 4 0
20 16 4 0
1 17 4 200
2 17 4 200
3 17 4 0
4 17 4 0
5 17 4 0
6 17 4 0
7 17 4 0
8 17 4 0
9 17 4 0
10 17 4 0
11 17 4 0
12 17 4 0
13 17 4 0
14 17 4 0
15 17 4 0
16 17 4 0
17 17 4 0
18 17 4 0
19 17 4 0
20 17 4 0
1 18 4 200
2 18 4 194
3 18 4 0
4 18 4 0
5 18 4 0
6 18 4 0
7 18 4 0
8 18 4 0
9 18 4 0
10 18 4 0
11 18 4 0
12 18 4 0
13 18 4 0
14 18 4 0
15 18 4 0
16 18 4 0
17 18 4 0
18 18 4 0
19 18 4 0
20 18 4 0
1 19 4 200
2 19 4 200
3 19 4 0
4 19 4 0
5 19 4 0
6 19 4 0
7 19 4 0
8 19 4 0
9 19 4 0
10 19 4 0
11 19 4 0
12 19 4 0
13 19 4 0
14 19 4 0
15 19 4 0
16 19 4 0
17 19 4 0
18 19 4 0
19 19 4 0
20 19 4 0
9 20 4 0
1 20 4 200
2 20 4 200
3 20 4 0
4 20 4 0
5 20 4 0
6 20 4 0
7 20 4 0
8 20 4 0
9 20 4 0
10 20 4 0
11 20 4 0
12 20 4 0
13 20 4 0
14 20 4 0
15 20 4 0
16 20 4 0
17 20 4 0
18 20 4 0
19 20 4 0
20 20 4 0
14 20 5 0
1 1 5 0
2 1 5 3
3 1 5 1
4 1 5 0
5 1 5 0
6 1 5 0
7 1 5 0
8 1 5 0
9 1 5 0
10 1 5 0
11 1 5 0
12 1 5 0
13 1 5 0
14 1 5 0
15 1 5 0
16 1 5 0
17 1 5 0
18 1 5 0
19 1 5 0
20 1 5 0
1 2 5 0
2 2 5 0
3 2 5 0
4 2 5 0
5 2 5 0
6 2 5 0
7 2 5 0
8 2 5 0
9 2 5 0
10 2 5 0
11 2 5 0
12 2 5 0
13 2 5 0
14 2 5 0
15 2 5 0
16 2 5 0
17 2 5 0
18 2 5 0
19 2 5 0
20 2 5 0
1 3 5 0
2 3 5 0
3 3 5 0
4 3 5 0
5 3 5 0
6 3 5 0
7 3 5 0
8 3 5 0
9 3 5 0
10 3 5 0
11 3 5 0
12 3 5 0
13 3 5 0
14 3 5 0
15 3 5 0
16 3 5 0
17 3 5 0
18 3 5 0
19 3 5 0
20 3 5 0
1 4 5 0
2 4 5 0
3 4 5 0
4 4 5 0
5 4 5 0
6 4 5 0
7 4 5 0
8 4 5 0
9 4 5 0
10 4 5 0
11 4 5 0
12 4 5 0
13 4 5 0
14 4 5 0
15 4 5 0
16 4 5 0
17 4 5 0
18 4 5 0
19 4 5 0
20 4 5 0
1 5 5 0
2 5 5 0
3 5 5 0
4 5 5 0
5 5 5 0
6 5 5 0
7 5 5 0
8 5 5 0
9 5 5 0
10 5 5 0
11 5 5 0
12 5 5 0
13 5 5 0
14 5 5 0
15 5 5 0
16 5 5 0
17 5 5 0
18 5 5 0
19 5 5 0
20 5 5 0
1 6 5 2
2 6 5 0
3 6 5 0
4 6 5 0
5 6 5 0
6 6 5 0
7 6 5 0
8 6 5 0
9 6 5 0
10 6 5 0
11 6 5 0
12 6 5 0
13 6 5 0
14 6 5 0
15 6 5 0
16 6 5 0
17 6 5 0
18 6 5 0
19 6 5 0
20 6 5 0
1 7 5 200
2 7 5 2
3 7 5 0
4 7 5 0
5 7 5 0
6 7 5 0
7 7 5 0
8 7 5 0
9 7 5 0
10 7 5 0
11 7 5 0
12 7 5 0
13 7 5 0
14 7 5 0

15 7 5 0
16 7 5 0
17 7 5 0
18 7 5 0
19 7 5 0
20 7 5 0
1 8 5 200
2 8 5 1
3 8 5 0
4 8 5 0
5 8 5 0
6 8 5 0
7 8 5 0
8 8 5 0
9 8 5 0
10 8 5 0
11 8 5 0
12 8 5 0
13 8 5 0
14 8 5 0
15 8 5 0
16 8 5 0
17 8 5 0
18 8 5 0
19 8 5 0
20 8 5 0
1 9 5 200
2 9 5 0
3 9 5 0
4 9 5 0
5 9 5 0
6 9 5 0
7 9 5 0
8 9 5 0
9 9 5 0
10 9 5 0
11 9 5 0
12 9 5 0
13 9 5 0
14 9 5 0
15 9 5 0
16 9 5 0
17 9 5 0
18 9 5 0
19 9 5 0
20 9 5 0
1 10 5 200
2 10 5 0
3 10 5 0
4 10 5 0
5 10 5 0
6 10 5 0
7 10 5 0
8 10 5 0
9 10 5 0
10 10 5 0
11 10 5 0
12 10 5 0
13 10 5 0
14 10 5 0
15 10 5 0
16 10 5 0
17 10 5 0
18 10 5 0
19 10 5 0
20 10 5 0
1 11 5 200
2 11 5 0
3 11 5 0
4 11 5 0
5 11 5 0
6 11 5 0
7 11 5 0
8 11 5 0
9 11 5 0
10 11 5 0
11 11 5 0
12 11 5 0
13 11 5 0
14 11 5 0
15 11 5 0
16 11 5 0
17 11 5 0
18 11 5 0
19 11 5 0
20 11 5 0
1 12 5 200
2 12 5 0
3 12 5 0
4 12 5 0
5 12 5 0
6 12 5 0
7 12 5 0
8 12 5 0
9 12 5 0
10 12 5 0
11 12 5 0
12 12 5 0
13 12 5 0
14 12 5 0
15 12 5 0
16 12 5 0
17 12 5 0
18 12 5 0
19 12 5 0
20 12 5 0
1 13 5 200
2 13 5 0
3 13 5 0
4 13 5 0
5 13 5 0
6 13 5 0
7 13 5 0
8 13 5 0
9 13 5 0
10 13 5 0
11 13 5 0
12 13 5 0
13 13 5 0
14 13 5 0
15 13 5 0
16 13 5 0
17 13 5 0
18 13 5 0
19 13 5 0
20 13 5 0
1 14 5 200
2 14 5 0
3 14 5 0
4 14 5 0
5 14 5 0
6 14 5 0
7 14 5 0
8 14 5 0
9 14 5 0
10 14 5 0
11 14 5 0
12 14 5 0
13 14 5 0
14 14 5 0
15 14 5 0
16 14 5 0
17 14 5 0
18 14 5 0
19 14 5 0
20 14 5 0
1 15 5 200
2 15 5 0
3 15 5 0
4 15 5 0
5 15 5 0
6 15 5 0
7 15 5 0
8 15 5 0
9 15 5 0
10 15 5 0
11 15 5 0
12 15 5 0
13 15 5 0
14 15 5 0
15 15 5 0
16 15 5 0
17 15 5 0
18 15 5 0
19 15 5 0
20 15 5 0
1 16 5 200
2 16 5 0
3 16 5 0
4 16 5 0
5 16 5 0
6 16 5 0
7 16 5 0
8 16 5 0
9 16 5 0
10 16 5 0
11 16 5 0
12 16 5 0
13 16 5 0
14 16 5 0
15 16 5 0
16 16 5 0
17 16 5 0
18 16 5 0
19 16 5 0
20 16 5 0
1 17 5 200
2 17 5 0
3 17 5 0
4 17 5 0
5 17 5 0
6 17 5 0
7 17 5 0
8 17 5 0
9 17 5 0
10 17 5 0
11 17 5 0
12 17 5 0
13 17 5 0
14 17 5 0
15 17 5 0
16 17 5 0
17 17 5 0
18 17 5 0
19 17 5 0
20 17 5 0
1 18 5 200
2 18 5 0
3 18 5 0
4 18 5 0
5 18 5 0
6 18 5 0
7 18 5 0
8 18 5 0
9 18 5 0
10 18 5 0
11 18 5 0
12 18 5 0
13 18 5 0
14 18 5 0
15 18 5 0
16 18 5 0
17 18 5 0
18 18 5 0
19 18 5 0
20 18 5 0
1 19 5 200
2 19 5 1
3 19 5 0
4 19 5 0
5 19 5 0
6 19 5 0
7 19 5 0
8 19 5 0
9 19 5 0
10 19 5 0
11 19 5 0
12 19 5 0
13 19 5 0
14 19 5 0
15 19 5 0
16 19 5 0
17 19 5 0
18 19 5 0
19 19 5 0
20 19 5 0
1 20 5 200
2 20 5 1
3 20 5 0
4 20 5 0
5 20 5 0
6 20 5 0
7 20 5 0
8 20 5 0
9 20 5 0
10 20 5 0
11 20 5 0
12 20 5 0
13 20 5 0
14 20 5 0
15 20 5 0
16 20 5 0
17 20 5 0
18 20 5 0
19 20 5 0
20 20 5 0
1 1 6 0
2 1 6 2
3 1 6 0
4 1 6 1
5 1 6 0
6 1 6 0
7 1 6 0
8 1 6 0
9 1 6 0
10 1 6 0
11 1 6 0
12 1 6 0
13 1 6 0
14 1 6 0
15 1 6 0
16 1 6 0
17 1 6 0
18 1 6 0
19 1 6 0
20 1 6 0
1 2 6 0
2 2 6 0
3 2 6 0
4 2 6 0
5 2 6 0
6 2 6 0
7 2 6 0
8 2 6 0
9 2 6 0
10 2 6 0
11 2 6 0
12 2 6 0
13 2 6 0
14 2 6 0
15 2 6 0
16 2 6 0
17 2 6 0
18 2 6 0
19 2 6 0
20 2 6 0
1 3 6 0
2 3 6 0
3 3 6 0
4 3 6 0
5 3 6 0
6 3 6 0
7 3 6 0
8 3 6 0
9 3 6 0
10 3 6 0
11 3 6 0
12 3 6 0
13 3 6 0
14 3 6 0
15 3 6 0
16 3 6 0
17 3 6 0
18 3 6 0
19 3 6 0
20 3 6 0
1 4 6 0
2 4 6 0
3 4 6 0
4 4 6 0
5 4 6 0
6 4 6 0
7 4 6 0
8 4 6 0
9 4 6 0
10 4 6 0
11 4 6 0
12 4 6 0
13 4 6 0
14 4 6 0
15 4 6 0
16 4 6 0
17 4 6 0
18 4 6 0
19 4 6 0
20 4 6 0
1 5 6 0
2 5 6 0
3 5 6 0
4 5 6 0
5 5 6 0
6 5 6 0
7 5 6 0
8 5 6 0
9 5 6 0
10 5 6 0
11 5 6 0
12 5 6 0
13 5 6 0
14 5 6 0
15 5 6 0
16 5 6 0
17 5 6 0
18 5 6 0
19 5 6 0
20 5 6 0
1 6 6 0
2 6 6 0
3 6 6 0
4 6 6 0
5 6 6 0
6 6 6 0
7 6 6 0
8 6 6 0
9 6 6 0
10 6 6 0
11 6 6 0
12 6 6 0
13 6 6 0
14 6 6 0
15 6 6 0
16 6 6 0
17 6 6 0
18 6 6 0
19 6 6 0
20 6 6 0
1 7 6 0
2 7 6 0
3 7 6 0
4 7 6 0
5 7 6 0
6 7 6 0
7 7 6 0
8 7 6 0
9 7 6 0
10 7 6 0
11 7 6 0
12 7 6 0
13 7 6 0
14 7 6 0
15 7 6 0
16 7 6 0
17 7 6 0
18 7 6 0
19 7 6 0
20 7 6 0
1 8 6 0
2 8 6 0
3 8 6 0
4 8 6 0
5 8 6 0
6 8 6 0
7 8 6 0
8 8 6 0
9 8 6 0
10 8 6 0
11 8 6 0
12 8 6 0
13 8 6 0
14 8 6 0
15 8 6 0
16 8 6 0
17 8 6 0
18 8 6 0
19 8 6 0
20 8 6 0
1 9 6 200
2 9 6 0
3 9 6 0
4 9 6 0
5 9 6 0
6 9 6 0
7 9 6 0
8 9 6 0
9 9 6 0
10 9 6 0
11 9 6 0
12 9 6 0
13 9 6 0
14 9 6 0
15 9 6 0
16 9 6 0
17 9 6 0
18 9 6 0
19 9 6 0
20 9 6 0
1 10 6 200
2 10 6 0
3 10 6 0
4 10 6 0
5 10 6 0
6 10 6 0
7 10 6 0
8 10 6 0
9 10 6 0
10 10 6 0
11 10 6 0
12 10 6 0
13 10 6 0
14 10 6 0
15 10 6 0
16 10 6 0
17 10 6 0
18 10 6 0
19 10 6 0
20 10 6 0
1 11 6 200
2 11 6 0
3 11 6 0
4 11 6 0
5 11 6 0
6 11 6 0
7 11 6 0
8 11 6 0
9 11 6 0
10 11 6 0
11 11 6 0
12 11 6 0
13 11 6 0
14 11 6 0
15 11 6 0
16 11 6 0
17 11 6 0
18 11 6 0
19 11 6 0
20 11 6 0
1 12 6 200
2 12 6 0
3 12 6 0
4 12 6 0
5 12 6 0
6 12 6 0
7 12 6 0
8 12 6 0
9 12 6 0
10 12 6 0
11 12 6 0
12 12 6 0
13 12 6 0
14 12 6 0
15 12 6 0
16 12 6 0
17 12 6 0
18 12 6 0
19 12 6 0
20 12 6 0
1 13 6 200
2 13 6 0
3 13 6 0
4 13 6 0
5 13 6 0
6 13 6 0
7 13 6 0
8 13 6 0
9 13 6 0
10 13 6 0
11 13 6 0
12 13 6 0
13 13 6 0
14 13 6 0
15 13 6 0
16 13 6 0
17 13 6 0
18 13 6 0
19 13 6 0
20 13 6 0
1 14 6 200
2 14 6 0
3 14 6 0
4 14 6 0
5 14 6 0
6 14 6 0
7 14 6 0
8 14 6 0
9 14 6 0
10 14 6 0
11 14 6 0
12 14 6 0
13 14 6 0
14 14 6 0
15 14 6 0
16 14 6 0
17 14 6 0
18 14 6 0
19 14 6 0
20 14 6 0
1 15 6 200
2 15 6 0
3 15 6 0
4 15 6 0
5 15 6 0
6 15 6 0
7 15 6 0
8 15 6 0
9 15 6 0
10 15 6 0
11 15 6 0
12 15 6 0
13 15 6 0
14 15 6 0
15 15 6 0
16 15 6 0
17 15 6 0
18 15 6 0
19 15 6 0
20 15 6 0
1 16 6 200
2 16 6 0
3 16 6 0
4 16 6 0
5 16 6 0
6 16 6 0
7 16 6 0
8 16 6 0
9 16 6 0
10 16 6 0
11 16 6 0
12 16 6 0
13 16 6 0
14 16 6 0
15 16 6 0
16 16 6 0
19 16 6 0
17 16 6 0
18 16 6 0
19 16 6 0
20 16 6 0
1 1 8 0
1 1 9 0
1 1 7 0
2 1 7 0
2 1 8 1
2 1 9 1
3 1 9 0
3 1 7 0
3 1 8 0
4 1 9 1
4 1 7 0
4 1 8 0
5 1 9 0
5 1 8 0
5 1 7 0
6 1 8 1
6 1 9 0
6 1 7 0
7 1 9 0
7 1 7 0
7 1 8 0
8 1 9 0
8 1 8 0
8 1 7 0
9 1 8 0
9 1 9 0
9 1 7 0
10 1 9 0
10 1 8 0
10 1 7 0
11 1 9 0
11 1 7 0
11 1 8 0
12 1 8 0
12 1 9 0
12 1 7 0
13 1 9 0
13 1 8 0
13 1 7 0
14 1 7 0
14 1 9 0
14 1 8 0
15 1 8 0
15 1 7 0
15 1 9 0
16 1 8 0
16 1 9 0
17 1 9 0
16 1 7 0
17 1 7 0
18 1 8 0
18 1 8 0
18 1 7 0
18 1 9 0
19 1 8 0
19 1 9 0
19 1 7 0
20 1 7 0
20 1 9 0
20 1 8 0
1 2 9 0
1 2 8 0
1 2 7 0
2 2 9 0
2 2 8 0
2 2 7 1
3 2 8 0
3 2 9 0
3 2 7 0
4 2 9 0
4 2 8 0
4 2 7 0
5 2 7 0
5 2 8 0
5 2 9 0
6 2 9 0
6 2 7 0
6 2 8 0
7 2 9 0
7 2 7 0
7 2 8 0
8 2 7 0
8 2 8 0
8 2 9 0
9 2 9 0
9 2 8 0
9 2 7 0
10 2 9 0
10 2 8 0
10 2 7 0
11 2 7 0
11 2 9 0
11 2 8 0
12 2 7 0
12 2 8 0
12 2 9 0
13 2 7 0
13 2 8 0
13 2 9 0
14 2 9 0
14 2 7 0
14 2 8 0
15 2 7 0
15 2 9 0
15 2 8 0
16 2 9 0
16 2 7 0
16 2 8 0
17 2 7 0
17 2 9 0
17 2 8 0
18 2 9 0
18 2 8 0
18 2 7 0
19 2 7 0
19 2 8 0
19 2 9 0
20 2 7 0
20 2 8 0
20 2 9 0
1 3 9 0
1 3 8 0
1 3 7 0
2 3 7 0
2 3 9 0
2 3 8 0
3 3 9 0
3 3 7 0
3 3 8 0
4 3 8 0
4 3 7 0
4 3 9 0
5 3 7 0
5 3 9 0
5 3 8 0
6 3 7 0
6 3 8 0
6 3 9 0
7 3 7 0
7 3 8 0
7 3 9 0
8 3 8 0
8 3 9 0
8 3 7 0
9 3 9 0
9 3 8 0
9 3 7 0
10 3 7 0
10 3 8 0
10 3 9 0
11 3 8 0
11 3 7 0
11 3 9 0
12 3 7 0
12 3 8 0
12 3 9 0
13 3 9 0
13 3 7 0
13 3 8 0
14 3 8 0
14 3 9 0
14 3 7 0
15 3 9 0
15 3 7 0
15 3 8 0
16 3 7 0
16 3 9 0
16 3 8 0
17 3 9 0
17 3 8 0
17 3 7 0
18 3 7 0
18 3 9 0
18 3 8 0
19 3 7 0
19 3 8 0
19 3 9 0
20 3 7 0
20 3 8 0
20 3 9 0
1 4 9 0
1 4 8 0
1 4 7 0
2 4 9 0
2 4 8 0
2 4 7 0
3 4 7 0
3 4 8 0
3 4 9 0
4 4 7 0
4 4 9 0
4 4 8 0
5 4 8 0
5 4 9 0
5 4 7 0
6 4 9 0
6 4 7 0
6 4 8 0
7 4 8 0
7 4 7 0
7 4 9 0
8 4 7 0
8 4 8 0
8 4 9 0
9 4 7 0
9 4 8 0
9 4 9 0
10 4 7 0
10 4 9 0
10 4 8 0
11 4 9 0
11 4 7 0
11 4 8 0
12 4 8 0
12 4 9 0
12 4 7 0
13 4 9 0
13 4 8 0
13 4 7 0
14 4 9 0
14 4 8 0
14 4 7 0
15 4 7 0
15 4 8 0
16 4 8 0
16 4 7 0
16 4 9 0
17 4 9 0
17 4 8 0
17 4 7 0
18 4 9 0
18 4 7 0
18 4 8 0
19 4 9 0
19 4 8 0
19 4 7 0
20 4 9 0
20 4 7 0
20 4 8 0
1 5 9 0
1 5 8 0
1 5 7 0
2 5 7 0
2 5 8 0
2 5 9 0
3 5 9 0
3 5 8 0
3 5 7 0
4 5 7 0
4 5 8 0
4 5 9 0
5 5 9 0
5 5 8 0
5 5 7 0
6 5 8 0
6 5 7 0
6 5 9 0
7 5 8 0
7 5 9 0
7 5 7 0
8 5 9 0
8 5 7 0
8 5 8 0
9 5 9 0
9 5 7 0
9 5 8 0
10 5 8 0
10 5 7 0
10 5 9 0
11 5 7 0
11 5 8 0
11 5 9 0
12 5 9 0
12 5 7 0
12 5 8 0
13 5 9 0
13 5 8 0
13 5 7 0
14 5 7 0
14 5 9 0
14 5 8 0
15 5 7 0
15 5 8 0
15 5 9 0
16 5 8 0
16 5 7 0
16 5 9 0
17 5 8 0
17 5 7 0
17 5 9 0
18 5 7 0
18 5 8 0

18 5 9 0
19 5 9 0
19 5 8 0
19 5 7 0
20 5 9 0
20 5 8 0
20 5 7 0
1 6 9 0
1 6 8 0
1 6 7 0
2 6 9 0
2 6 8 0
2 6 7 0
3 6 9 0
3 6 7 0
3 6 8 0
4 6 7 0
4 6 8 0
4 6 9 0
5 6 9 0
5 6 8 0
5 6 7 0
6 6 7 0
6 6 8 0
6 6 9 0
7 6 9 0
7 6 8 0
7 6 7 0
8 6 7 0
8 6 8 0
8 6 9 0
9 6 7 0
9 6 9 0
9 6 8 0
10 6 8 0
10 6 7 0
10 6 9 0
11 6 7 0
11 6 9 0
11 6 8 0
12 6 7 0
12 6 8 0
12 6 9 0
13 6 7 0
13 6 8 0
13 6 9 0
14 6 8 0
14 6 9 0
14 6 7 0
15 6 8 0
15 6 7 0
15 6 9 0
16 6 9 0
16 6 7 0
16 6 8 0
17 6 9 0
17 6 7 0
17 6 8 0
18 6 8 0
18 6 7 0
18 6 9 0
19 6 7 0
19 6 9 0
19 6 8 0
20 6 8 0
20 6 9 0
20 6 7 0
1 7 9 0
1 7 8 0
1 7 7 0
2 7 7 0
2 7 9 0
2 7 8 0
3 7 9 0
3 7 8 0
3 7 7 0
4 7 9 0
4 7 7 0
4 7 8 0
5 7 9 0
5 7 8 0
5 7 7 0
6 7 7 0
6 7 8 0
6 7 9 0
7 7 9 0
7 7 8 0
7 7 7 0
8 7 9 0
8 7 7 0
8 7 8 0
9 7 8 0
9 7 7 0
9 7 9 0
10 7 8 0
10 7 7 0
10 7 9 0
11 7 8 0
11 7 9 0
11 7 7 0
12 7 7 0
12 7 8 0
12 7 9 0
13 7 8 0
13 7 9 0
13 7 7 0
14 7 7 0
14 7 9 0
14 7 8 0
15 7 8 0
15 7 9 0
15 7 7 0
16 7 8 0
16 7 9 0
16 7 7 0
17 7 7 0
17 7 9 0
17 7 8 0
18 7 7 0
18 7 9 0
18 7 8 0
19 7 7 0
19 7 9 0
19 7 8 0
20 7 7 0
20 7 9 0
20 7 8 0
1 8 9 0
1 8 8 0
1 8 7 0
2 8 9 0
2 8 7 0
2 8 8 0
3 8 7 0
3 8 9 0
3 8 8 0
4 8 9 0
4 8 8 0
4 8 7 0
5 8 9 0
5 8 7 0
5 8 8 0
6 8 7 0
6 8 8 0
6 8 9 0
7 8 9 0
7 8 8 0
7 8 7 0
8 8 9 0
8 8 8 0
8 8 7 0
9 8 7 0
9 8 8 0
9 8 9 0
10 8 9 0
10 8 8 0
10 8 7 0
11 8 9 0
11 8 8 0
11 8 7 0
12 8 8 0
12 8 9 0
12 8 7 0
13 8 7 0
13 8 8 0
13 8 9 0
14 8 7 0
14 8 9 0
14 8 8 0
15 8 9 0
15 8 8 0
15 8 7 0
16 8 7 0
16 8 9 0
16 8 8 0
17 8 9 0
17 8 7 0
17 8 8 0
18 8 8 0
18 8 9 0
18 8 7 0
19 8 7 0
19 8 9 0
19 8 8 0
20 8 8 0
20 8 7 0
20 8 9 0
1 9 9 0
1 9 8 0
1 9 7 0
2 9 7 0
2 9 9 0
2 9 8 0
3 9 8 0
3 9 9 0
3 9 7 0
4 9 7 0
4 9 9 0
4 9 8 0
5 9 7 0
5 9 9 0
5 9 8 0
6 9 7 0
6 9 9 0
6 9 8 0
7 9 7 0
7 9 8 0
7 9 9 0
8 9 7 0
8 9 9 0
8 9 8 0
9 9 9 0
9 9 8 0
9 9 7 0
10 9 8 0
10 9 9 0
10 9 7 0
11 9 7 0
11 9 8 0
11 9 9 0
12 9 9 0
12 9 7 0
12 9 8 0
13 9 9 0
13 9 7 0
13 9 8 0
14 9 8 0
14 9 9 0
14 9 7 0
15 9 9 0
15 9 7 0
15 9 8 0
16 9 9 0
16 9 7 0
16 9 8 0
17 9 7 0
17 9 8 0
17 9 9 0
18 9 7 0
18 9 9 0
18 9 8 0
19 9 8 0
19 9 9 0
19 9 7 0
20 9 9 0
20 9 8 0
20 9 7 0
1 10 9 0
1 10 8 0
1 10 7 0
2 10 7 0
2 10 8 0
2 10 9 0
3 10 9 0
3 10 7 0
3 10 8 0
4 10 7 0
4 10 8 0
4 10 9 0
5 10 7 0
5 10 9 0
5 10 8 0
6 10 8 0
6 10 9 0
6 10 7 0
7 10 9 0
7 10 7 0
7 10 8 0
8 10 7 0
8 10 9 0
8 10 8 0
9 10 7 0
9 10 9 0
9 10 8 0
10 10 7 0
10 10 8 0
10 10 9 0
11 10 7 0
11 10 8 0
11 10 9 0
12 10 7 0
12 10 9 0
12 10 8 0
13 10 7 0
13 10 8 0
13 10 9 0
14 10 8 0
14 10 7 0
14 10 9 0
15 10 9 0
15 10 7 0
15 10 8 0
16 10 7 0
16 10 9 0
16 10 8 0
17 10 7 0
17 10 8 0
17 10 9 0
18 10 7 0
18 10 8 0
18 10 9 0
19 10 9 0
19 10 8 0
19 10 7 0
20 10 9 0
20 10 8 0
20 10 7 0
1 11 9 0
1 11 8 0
1 11 7 13
2 11 7 0
2 11 9 0
2 11 8 0
3 11 7 0
3 11 8 0
3 11 9 0
4 11 8 0
4 11 7 0
4 11 9 0
5 11 7 0
5 11 9 0
5 11 8 0
6 11 7 0
6 11 9 0
6 11 8 0
7 11 7 0
7 11 8 0
7 11 9 0
8 11 7 0
8 11 9 0
8 11 8 0
9 11 7 0
9 11 9 0
9 11 8 0
10 11 9 0
10 11 7 0
10 11 8 0
11 11 7 0
11 11 9 0
11 11 8 0
12 11 9 0
12 11 8 0
12 11 7 0
13 11 9 0
13 11 7 0
13 11 8 0
14 11 7 0
14 11 8 0
14 11 9 0
15 11 7 0
15 11 8 0
15 11 9 0
16 11 8 0
16 11 9 0
16 11 7 0
17 11 9 0
17 11 7 0
17 11 8 0
18 11 9 0
18 11 8 0
18 11 7 0
19 11 9 0
19 11 8 0
19 11 7 0
20 11 9 0
20 11 8 0
20 11 7 0
1 12 9 0
1 12 8 0
1 12 7 200
2 12 7 0
2 12 9 0
2 12 8 0
3 12 9 0
3 12 8 0
3 12 7 0
4 12 7 0
4 12 9 0
4 12 8 0
5 12 7 0
5 12 9 0
5 12 8 0
6 12 7 0
6 12 9 0
6 12 8 0
7 12 7 0
7 12 9 0
7 12 8 0
8 12 9 0
8 12 7 0
8 12 8 0
9 12 9 0
9 12 7 0
9 12 8 0
10 12 7 0
10 12 8 0
10 12 9 0
11 12 7 0
11 12 8 0
11 12 9 0
12 12 7 0
12 12 9 0
12 12 8 0
13 12 9 0
13 12 7 0
13 12 8 0
14 12 9 0
14 12 8 0
14 12 7 0
15 12 7 0
15 12 9 0
15 12 8 0
16 12 7 0
16 12 9 0
16 12 8 0
17 12 7 0
17 12 9 0
17 12 8 0
18 12 7 0
18 12 8 0
18 12 9 0
19 12 9 0
19 12 7 0
19 12 8 0
20 12 8 0
20 12 7 0
20 12 9 0
1 13 9 0
1 13 8 0
1 13 7 200
2 13 8 0
2 13 9 0
2 13 7 0
3 13 7 0
3 13 9 0
3 13 8 0
4 13 7 0
4 13 9 0
4 13 8 0
5 13 9 0
5 13 7 0
5 13 8 0
6 13 9 0
6 13 8 0
6 13 7 0
7 13 7 0
7 13 8 0
7 13 9 0
8 13 8 0
8 13 9 0
8 13 7 0
9 13 7 0
9 13 9 0
9 13 8 0
10 13 7 0
10 13 8 0
10 13 9 0
11 13 7 0
11 13 9 0
11 13 8 0
12 13 8 0
12 13 9 0
12 13 7 0
13 13 9 0
13 13 8 0
13 13 7 0
14 13 9 0
14 13 8 0
14 13 7 0
15 13 8 0
15 13 7 0
15 13 9 0
16 13 9 0
16 13 8 0
16 13 7 0
17 13 8 0
17 13 9 0
17 13 7 0
18 13 9 0
18 13 8 0
18 13 7 0
19 13 9 0
19 13 7 0
19 13 8 0
20 13 7 0
20 13 8 0
20 13 9 0
1 14 7 200
1 14 9 0
1 14 8 9
2 14 7 0
2 14 9 0
2 14 8 0
3 14 8 0
3 14 7 0
3 14 9 0
4 14 7 0
4 14 8 0
4 14 9 0
5 14 7 0
5 14 8 0
5 14 9 0
6 14 7 0
6 14 9 0
6 14 8 0
7 14 7 0
7 14 8 0
7 14 9 0
8 14 9 0
8 14 8 0
8 14 7 0
9 14 7 0
9 14 9 0
9 14 8 0
10 14 7 0
10 14 9 0
10 14 8 0
11 14 9 0
11 14 8 0
11 14 7 0
12 14 7 0
12 14 8 0
12 14 9 0
13 14 8 0
13 14 9 0
13 14 7 0
14 14 8 0
14 14 9 0
14 14 7 0
15 14 8 0
15 14 9 0
15 14 7 0
16 14 7 0
16 14 9 0
16 14 8 0
17 14 9 0
17 14 8 0
17 14 7 0
18 14 9 0
18 14 8 0
18 14 7 0
19 14 8 0
19 14 9 0
19 14 7 0
20 14 8 0
20 14 7 0
20 14 9 0
1 15 7 200
1 15 9 0
1 15 8 200
2 15 8 0
2 15 9 0
2 15 7 0
3 15 7 0
3 15 8 0
3 15 9 0
4 15 8 0
4 15 7 0
4 15 9 0
5 15 7 0
5 15 8 0
5 15 9 0
6 15 8 0
6 15 9 0
6 15 7 0
7 15 7 0
7 15 9 0
7 15 8 0
8 15 7 0
8 15 9 0
8 15 8 0
9 15 9 0
9 15 7 0
9 15 8 0
10 15 7 0
10 15 8 0
10 15 9 0
11 15 9 0
11 15 7 0
11 15 8 0
12 15 8 0
12 15 7 0
12 15 9 0
13 15 7 0
13 15 8 0
13 15 9 0
14 15 8 0
14 15 7 0
14 15 9 0
15 15 9 0
15 15 8 0
15 15 7 0
16 15 9 0
16 15 8 0
16 15 7 0
17 15 8 0
17 15 7 0
17 15 9 0
18 15 9 0
18 15 8 0
18 15 7 0
19 15 8 0
19 15 9 0
19 15 7 0
20 15 8 0
20 15 9 0
20 15 7 0
1 16 7 200
1 16 9 0
1 16 8 200
2 16 9 0
2 16 7 0
2 16 8 0
3 16 9 0
3 16 8 0
3 16 7 0
4 16 7 0
4 16 9 0
4 16 8 0
5 16 7 0
5 16 9 0
5 16 8 0
6 16 7 0
6 16 9 0
6 16 8 0
7 16 7 0
7 16 8 0
7 16 9 0
8 16 9 0
8 16 7 0
8 16 8 0
9 16 9 0
9 16 8 0
9 16 7 0
10 16 7 0
10 16 8 0
10 16 9 0
11 16 8 0
11 16 9 0
11 16 7 0
12 16 7 0
12 16 8 0
12 16 9 0
13 16 9 0
13 16 8 0
13 16 7 0
14 16 9 0
14 16 7 0
14 16 8 0
15 16 9 0
15 16 8 0
15 16 7 0
16 16 9 0
16 16 7 0
16 16 8 0
17 16 9 0
17 16 7 0
17 16 8 0
18 16 9 0
18 16 8 0
18 16 7 0
19 16 8 0
19 16 7 0
19 16 9 0
20 16 9 0
20 16 8 0
20 16 7 0
1 17 7 200
1 17 8 200
1 17 9 1
2 17 7 0
2 17 9 0
2 17 8 0
3 17 7 0
3 17 9 0
3 17 8 0
4 17 9 0
4 17 7 0
4 17 8 0
5 17 7 0
5 17 9 0
5 17 8 0
6 17 9 0
6 17 7 0
6 17 8 0
7 17 9 0
7 17 8 0
7 17 7 0
8 17 9 0
8 17 8 0
8 17 7 0
9 17 9 0
9 17 7 0
9 17 8 0
10 17 9 0
10 17 7 0
10 17 8 0
11 17 9 0
11 17 7 0
11 17 8 0
12 17 9 0
12 17 7 0
12 17 8 0
13 17 7 0
13 17 8 0
13 17 9 0
14 17 9 0
14 17 8 0
14 17 7 0
15 17 9 0
15 17 7 0
15 17 8 0
16 17 9 0
16 17 8 0
16 17 7 0
17 17 9 0
17 17 8 0
17 17 7 0
18 17 9 0
18 17 8 0
18 17 7 0
19 17 8 0
19 17 7 0
20 17 9 0
20 17 7 0
20 17 8 0
1 18 7 200
1 18 8 200
1 18 9 22
2 18 9 0
2 18 7 0
2 18 8 0
3 18 8 0
3 18 9 0
3 18 7 0
4 18 9 0
4 18 7 0
4 18 8 0
5 18 7 0
5 18 8 0
5 18 9 0
6 18 9 0
6 18 8 0
6 18 7 0
7 18 8 0
7 18 7 0
7 18 9 0
8 18 8 0
8 18 9 0
8 18 7 0
9 18 7 0
9 18 8 0
9 18 9 0
10 18 9 0
10 18 7 0
10 18 8 0
11 18 9 0
11 18 8 0
11 18 7 0
12 18 8 0
12 18 9 0
12 18 7 0
13 18 9 0
13 18 8 0
13 18 7 0
14 18 9 0
14 18 8 0
14 18 7 0
15 18 9 0
15 18 8 0
15 18 7 0
16 18 7 0
16 18 9 0
16 18 8 0
17 18 7 0
17 18 9 0
17 18 8 0
18 18 9 0
18 18 7 0
18 18 8 0
19 18 9 0
19 18 8 0
19 18 7 0
20 18 9 0
20 18 8 0
20 18 7 0
1 19 7 200
1 19 8 200
1 19 9 200
2 19 8 0
2 19 7 0
2 19 9 0
3 19 7 0
3 19 8 0
3 19 9 0
4 19 7 0
4 19 9 0
4 19 8 0
5 19 8 0
5 19 7 0
5 19 9 0
6 19 8 0
6 19 9 0
6 19 7 0
7 19 7 0
7 19 8 0
7 19 9 0
8 19 9 0
8 19 8 0
8 19 7 0
9 19 9 0
9 19 7 0
9 19 8 0
10 19 9 0
10 19 7 0
10 19 8 0
11 19 7 0
11 19 9 0
11 19 8 0
12 19 9 0
12 19 8 0
12 19 7 0
13 19 9 0
13 19 7 0
13 19 8 0
14 19 9 0
14 19 7 0
14 19 8 0
15 19 9 0
15 19 8 0
15 19 7 0
16 19 7 0
16 19 9 0
16 19 8 0
17 19 9 0
17 19 8 0
17 19 7 0
18 19 9 0
18 19 8 0
18 19 7 0
19 19 9 0
19 19 8 0
19 19 7 0
20 19 8 0
20 19 9 0
20 19 7 0
1 20 7 200
1 20 8 200
1 20 9 200
2 20 8 0
2 20 7 0
2 20 9 0
3 20 9 0
3 20 7 0
3 20 8 0
4 20 7 0
4 20 9 0
4 20 8 0
5 20 7 0
5 20 9 0
5 20 8 0
6 20 7 0
6 20 9 0
6 20 8 0
7 20 8 0
7 20 7 0
7 20 9 0
8 20 7 0
8 20 8 0
8 20 9 0
9 20 7 0
9 20 9 0
9 20 8 0
10 20 8 0
10 20 9 0
10 20 7 0
11 20 9 0
11 20 7 0
11 20 8 0

12 20 7 0
12 20 9 0
12 20 8 0
13 20 8 0
13 20 7 0
13 20 9 0
14 20 9 0
14 20 8 0
14 20 7 0
15 20 9 0
15 20 8 0
15 20 7 0
16 20 9 0
16 20 8 0
16 20 7 0
17 20 9 0
17 20 7 0
17 20 8 0
18 20 9 0
18 20 7 0
18 20 8 0
19 20 9 0
19 20 8 0
19 20 7 0
20 20 9 0
20 20 8 0
20 20 7 0
1 1 10 0
1 2 10 0
1 8 10 0
1 3 10 0
1 13 10 0
1 4 10 0
1 9 10 0
1 5 10 0
1 6 10 0
1 14 10 0
1 10 10 0
1 7 10 0
1 11 10 0
1 15 10 0
1 16 10 0
1 12 10 0
1 18 10 0
1 17 10 0
1 19 10 0
1 20 10 0
2 1 10 0
2 2 10 0
2 3 10 0
2 8 10 0
2 4 10 0
2 13 10 0
2 5 10 0
2 9 10 0
2 6 10 0
2 14 10 0
2 10 10 0
2 7 10 0
2 11 10 0
2 15 10 0
2 16 10 0
2 12 10 0
2 18 10 0
2 17 10 0
2 19 10 0
2 20 10 0
3 1 10 0
3 2 10 0
3 3 10 0
3 8 10 0
3 13 10 0
3 4 10 0
3 9 10 0
3 5 10 0
3 6 10 0
3 14 10 0
3 10 10 0
3 7 10 0
3 11 10 0
3 15 10 0
3 16 10 0
3 12 10 0
3 18 10 0
3 17 10 0
3 19 10 0
3 20 10 0
4 1 10 0
4 2 10 0
4 3 10 0
4 8 10 0
4 4 10 0
4 13 10 0
4 9 10 0
4 5 10 0
4 6 10 0
4 14 10 0
4 10 10 0
4 7 10 0
4 11 10 0
4 15 10 0
4 16 10 0
4 12 10 0
4 18 10 0
4 17 10 0
4 19 10 0
4 20 10 0
5 1 10 0
5 2 10 0
5 3 10 0
5 8 10 0
5 4 10 0
5 13 10 0
5 9 10 0
5 5 10 0
5 6 10 0
5 14 10 0
5 10 10 0
5 7 10 0
5 11 10 0
5 15 10 0
5 16 10 0
5 12 10 0
5 18 10 0
5 17 10 0
5 19 10 0
5 20 10 0
6 1 10 0
6 2 10 0
6 3 10 0
6 8 10 0
6 13 10 0
6 4 10 0
6 9 10 0
6 5 10 0
6 6 10 0
6 14 10 0
6 10 10 0
6 7 10 0
6 11 10 0
6 15 10 0
6 16 10 0
6 12 10 0
6 18 10 0
6 17 10 0
6 19 10 0
6 20 10 0
7 1 10 0
7 2 10 0
7 3 10 0
7 8 10 0
7 13 10 0
7 4 10 0
7 9 10 0
7 5 10 0
7 6 10 0
7 14 10 0
7 10 10 0
7 7 10 0
7 11 10 0
7 15 10 0
7 16 10 0
7 12 10 0
7 18 10 0
7 17 10 0
7 19 10 0
7 20 10 0
8 1 10 0
8 2 10 0
8 3 10 0
8 8 10 0
8 13 10 0
8 4 10 0
8 9 10 0
8 5 10 0
8 6 10 0
8 14 10 0
8 10 10 0
8 7 10 0
8 11 10 0
8 15 10 0
8 16 10 0
8 12 10 0
8 18 10 0
8 17 10 0
8 19 10 0
8 20 10 0
9 1 10 0
9 2 10 0
9 3 10 0
9 8 10 0
9 13 10 0
9 4 10 0
9 9 10 0
9 5 10 0
9 6 10 0
9 14 10 0
9 10 10 0
9 7 10 0
9 11 10 0
9 15 10 0
9 16 10 0
9 12 10 0
9 18 10 0
9 17 10 0
9 19 10 0
9 20 10 0
10 1 10 0
10 2 10 0
10 3 10 0
10 8 10 0
10 4 10 0
10 13 10 0
10 9 10 0
10 5 10 0
10 6 10 0
10 14 10 0
10 10 10 0
10 7 10 0
10 11 10 0
10 15 10 0
10 16 10 0
10 12 10 0
10 18 10 0
10 17 10 0
10 19 10 0
10 20 10 0
11 1 10 0
11 2 10 0
11 3 10 0
11 8 10 0
11 4 10 0
11 13 10 0
11 9 10 0
11 5 10 0
11 6 10 0
11 14 10 0
11 10 10 0
11 7 10 0
11 11 10 0
11 15 10 0
11 16 10 0
11 12 10 0
11 18 10 0
11 17 10 0
11 19 10 0
11 20 10 0
12 1 10 0
12 2 10 0
12 3 10 0
12 8 10 0
12 4 10 0
12 13 10 0
12 9 10 0
12 5 10 0
12 6 10 0
12 14 10 0
12 10 10 0
12 7 10 0
12 11 10 0
12 15 10 0
12 16 10 0
12 12 10 0
12 18 10 0
12 17 10 0
12 19 10 0
12 20 10 0
13 1 10 0
13 2 10 0
13 3 10 0
13 8 10 0
13 4 10 0
13 13 10 0
13 9 10 0
13 5 10 0
13 6 10 0
13 14 10 0
13 10 10 0
13 7 10 0
13 11 10 0
13 15 10 0
13 16 10 0
13 12 10 0
13 18 10 0
13 17 10 0
13 19 10 0
13 20 10 0
14 1 10 0
14 2 10 0
14 3 10 0
14 8 10 0
14 4 10 0
14 13 10 0
14 9 10 0
14 5 10 0
14 6 10 0
14 14 10 0
14 10 10 0
14 7 10 0
14 11 10 0
14 15 10 0
14 16 10 0
14 12 10 0
14 18 10 0
14 17 10 0
14 19 10 0
14 20 10 0
15 1 10 0
15 2 10 0
15 3 10 0
15 8 10 0
15 4 10 0
15 13 10 0
15 9 10 0
15 5 10 0
15 6 10 0
15 14 10 0
15 10 10 0
15 7 10 0
15 11 10 0
15 15 10 0
15 16 10 0
15 12 10 0
15 18 10 0
15 17 10 0
15 19 10 0
15 20 10 0
16 1 10 0
16 2 10 0
16 3 10 0
16 8 10 0
16 4 10 0
16 13 10 0
16 9 10 0
16 5 10 0
16 6 10 0
16 14 10 0
16 10 10 0
16 7 10 0
16 11 10 0
16 15 10 0
16 16 10 0
16 12 10 0
16 18 10 0
16 17 10 0
16 19 10 0
16 20 10 0
17 1 10 0
17 2 10 0
17 3 10 0
17 8 10 0
17 4 10 0
17 13 10 0
17 9 10 0
17 5 10 0
17 6 10 0
17 14 10 0
17 10 10 0
17 7 10 0
17 11 10 0
17 15 10 0
17 16 10 0
17 12 10 0
17 18 10 0
17 17 10 0
17 19 10 0
17 20 10 0
18 1 10 0
18 2 10 0
18 3 10 0
18 8 10 0
18 4 10 0
18 13 10 0
18 9 10 0
18 5 10 0
18 6 10 0
18 14 10 0
18 10 10 0
18 7 10 0
18 11 10 0
18 15 10 0
18 16 10 0
18 12 10 0
18 18 10 0
18 17 10 0
18 19 10 0
18 20 10 0
19 1 10 0
19 2 10 0
19 3 10 0
19 8 10 0
19 4 10 0
19 13 10 0
19 9 10 0
19 5 10 0
19 6 10 0
19 14 10 0
19 10 10 0
19 7 10 0
19 11 10 0
19 15 10 0
19 16 10 0
19 12 10 0
19 18 10 0
19 17 10 0
19 19 10 0
19 20 10 0
20 1 10 0
20 2 10 0
20 3 10 0
20 8 10 0
20 4 10 0
20 13 10 0
20 9 10 0
20 5 10 0
20 6 10 0
20 14 10 0
20 10 10 0
20 7 10 0
20 11 10 0
20 15 10 0
20 16 10 0
20 12 10 0
20 18 10 0
20 17 10 0
20 19 10 0
20 20 10 0
1 19 6 200
1 20 6 200
1 17 6 200
1 18 6 200
2 17 6 0
2 19 6 0
2 20 6 0
2 18 6 0
3 17 6 0
3 19 6 0
3 20 6 0
3 18 6 0
4 17 6 0
4 19 6 0
4 20 6 0
4 18 6 0
5 17 6 0
5 19 6 0
5 20 6 0
5 18 6 0
6 17 6 0
6 19 6 0
6 20 6 0
6 18 6 0
7 17 6 0
7 19 6 0
7 20 6 0
7 18 6 0
8 17 6 0
8 19 6 0
8 20 6 0
8 18 6 0
9 17 6 0
9 19 6 0
9 20 6 0
9 18 6 0
10 17 6 0
10 19 6 0
10 20 6 0
10 18 6 0
11 17 6 0
11 19 6 0
11 20 6 0
11 18 6 0
12 17 6 0
12 19 6 0
12 20 6 0
12 18 6 0
13 17 6 0
13 19 6 0
13 20 6 0
13 18 6 0
14 17 6 0
14 19 6 0
14 20 6 0
14 18 6 0
15 17 6 0
15 19 6 0
15 20 6 0
15 18 6 0
16 17 6 0
16 19 6 0
16 20 6 0
16 18 6 0
17 17 6 0
17 19 6 0
17 20 6 0
17 18 6 0
18 17 6 0
18 19 6 0
18 20 6 0
18 18 6 0
19 17 6 0
19 19 6 0
19 20 6 0
19 18 6 0
20 17 6 0
20 19 6 0
20 20 6 0
20 18 6 0
1 8 3 200
2 8 3 200
3 8 3 180
4 8 3 170
5 8 3 150
6 8 3 120
7 8 3 99
8 8 3 37
9 8 3 30

6.2 Program to add attacker.

6.2.1 Main Program.

```
#include "3kprotocol.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#include <mpi.h>
int EPOCH_LIMIT = 200000;
int SYNCHRONISATION_THRESHOLD = 2000;
int comm_sz;
int my_rank;
int values[10];
int values2[2];
int textOff = 0;


void printOutNetworks(struct NeuralNetwork, struct NeuralNetwork, struct NeuralNetwork*);
// values[0] = k
// values[1] = n
// values[2] = l
// values[3] = how many tests
// values[4] = how many threads
// values[5] = char length of max int
// values[6] = time
// values[7] = sync max
// values[8] = epoch max
// values[9] = display neural networks 1 if we would like to 0 if we would like to hide.
// values[10] = #times out of 200 synchronized

int main(int argc, char *argv[])
{
  values[9] = 0; // by default hide neural networks
  values[3] = 1; // by default set the amount of tests to 1.
  values[8] = 0; // by default set the epoch max to 0.
  MPI_Init(NULL, NULL); // initialize Mpi
  MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  // if my rank is 0
  if(my_rank == 0)
  {
    // if we have an argument with the program then turn the text off else display the text.
    if(argc==2)
    {
      textOff = 1;
    }
    if(textOff==0) printf("Please enter your value for k\n");
    scanf("%d", &values[0]);
    if(textOff==0) printf("Please enter your value for n\n");
    scanf("%d", &values[1]);
    if(textOff==0) printf("Please enter your value for l\n");
    scanf("%d", &values[2]);
    if(textOff==0) printf("Please enter how many tests your would like to run\n");
    scanf("%d", &values[3]);
    if(textOff==0) printf("Please enter how many openmp threads you would like to run.\n");
    scanf("%d", &values[4]);
    if(textOff==0) printf("Running pre-tests to work out normal synchronization threshold as well
      as normal epoch's\n\n");
    // get the amount of chars that the max L will use.
    values[5] = numPlaces(values[2])+1;
  }
  // broadcast all values.
  MPI_Bcast(&values, 11, MPI_INT, 0, MPI_COMM_WORLD);
  // for 40 tests perform a serial version of the kkk algoriothm without an attacker
  if(my_rank==0)
  {
    for(int j = 0; j < 40; j++)
    {
      // set a random seed for srand depending on iteration and time.
      srand(time(NULL)+j);
      // create a random neural network for A and B.
      struct NeuralNetwork neuralNetA = constructNeuralNetwork(values[0], values[1], values[2]);
      struct NeuralNetwork neuralNetB = constructNeuralNetwork(values[0], values[1], values[2]);
      // allocate space for inputs.
      int** inputs = malloc(sizeof (int*) * values[0]);
      for (int i = 0; i < values[0]; i++)
      {
        inputs[i] = malloc(sizeof (int) * values[1]);
      }
      // get random inputs.
      getRandomInputs(inputs, values[0], values[1]);
      // set out current sync maximum/ epoch maximum to 0
      int syncCheck = 0;
      int epochCheck = 0;
      // run the kkk algorithm without any attackers.
      bool status = runKKKProtocolWithoutAttaacker(neuralNetA, neuralNetB, inputs, values[0],
        values[1], values[2], SYNCHRONISATION_THRESHOLD, EPOCH_LIMIT, &syncCheck, &epochCheck);
      // free all inputs.
      for (int i = 0; i < values[0]; i++)
      {
        free(inputs[i]);
      }
      free(inputs);
      // if the networks synchronized then set max epoch and max synch's
      if(status == true && compareNetworks(neuralNetA, neuralNetB, values[0], values[1]))
      {
        if(values[7]<syncCheck)
        {
          values[7]=syncCheck;
        }
        if(values[8]<epochCheck)
        {
          values[8]=epochCheck;
        }
      }
```

```
    // free neural networks.
    freeMemoryForNetwork(neuralNetA, values[0], values[1]);
    freeMemoryForNetwork(neuralNetB, values[0], values[1]);
  }
}
// if my rank is 0 and we would like to print instructions
if(my_rank == 0 && textOff == 0)
{
  // tell the user we have finished running pretests and tell them the max epoch's and
    synchronization needed, ask them that
  // this is ok or if they woud like to use their own values.
  printf("Finished running pre-tests, synchronization threshold = %d, epoch threshold = %d\n",
    values[7]+1,values[8]+20);
  printf("If you would like to use these values type \"1\" for yes else type \"0\" for no.\n");
  int reply;
  scanf("%d", &reply);
  if(reply==0)
  {
    printf("Please state your synchronization threshold\n");
    scanf("%d", &values[7]);
    printf("Please state your epoch threshold\n");
    scanf("%d", &values[8]);
  }
  printf("Would you like to view the neural network weights before the process and after, type
    \"1\" for yes and \"0\" for no.\n");
  scanf("%d", &values[9]);
  printf("\n");
}
// for the amount of tests we would like to carry out.
for(int j = 0; j < values[3]; j++)
{
  // if my_rank is 0 get the shared time value.
  if(my_rank==0)
  {
    values[6] = (int)time(NULL)+j;
  }
  // share the time value.
  MPI_Bcast(&values, 11, MPI_INT, 0, MPI_COMM_WORLD);
  // if we wouldn't like to run any tests break.
  if(values[3]==0)
  {
    break;
  }
  // set our seed to our shared time value plus the test number.
  srand(values[6]+j);
  // create an array to store all neural network c's
  struct NeuralNetwork* neuralNetC = malloc(sizeof(struct NeuralNetwork)*values[4]);
  // construct neural net A and B.
  struct NeuralNetwork neuralNetA = constructNeuralNetwork(values[0], values[1], values[2]);
  struct NeuralNetwork neuralNetB = constructNeuralNetwork(values[0], values[1], values[2]);
  // in parallel generate all neural networks C.
  #pragma omp parallel for num_threads(values[4])
  for(int i = 0; i < values[4]; i++)
  {
    srand(rand()+(my_rank+1));
    srand(rand()+(omp_get_thread_num()+1));
    neuralNetC[i] = constructNeuralNetwork(values[0], values[1], values[2]);
  }
  // if we would like to, print neural networks print neural network A B and C in order of
    ranks.
  if(values[9]==1)
  {
    if(my_rank==0) printf("---------- Test %d ----------\nBefore\n", j);
    printOutNetworks(neuralNetA,neuralNetB,neuralNetC);
  }
  // set our random seed back to the shared seed so all inputs genrated are the same.
  srand(values[6]+j+5);
  // malloc space for our input values.
  int** inputs = malloc(sizeof (int*) * values[0]);
  for (int i = 0; i < values[0]; i++)
  {
    inputs[i] = malloc(sizeof (int) * values[1]);
  }
  // generate the random inputs.
  getRandomInputs(inputs, values[0], values[1]);
  // run the geometric attack on A, B and with maxSync+20 and maxEpoch+20.
  bool status = runGeometricAttackKKKProtocolParallel(neuralNetA, neuralNetB, neuralNetC,
    inputs, values[0], values[1], values[2], (values[7]+20), (values[8]+20));
  // if we would like to, print neural networks A, B and C in order of ranks.
  if(values[9]==1)
  {
    if(my_rank==0) printf("After\n");
    printOutNetworks(neuralNetA,neuralNetB,neuralNetC);
  }
  // set a value to sum all threads that managed to successfully attack A and B.
  int checkValue = 0;
  // for each neural network C check if C is the same as A,B and C, if they are then sum them
    for our omp group, afterwards free C.
  #pragma omp parallel for reduction(+:checkValue) num_threads(values[4])
  for(int i = 0; i < values[4]; i++)
  {
    if(status==true&&compareNetworks(neuralNetA,neuralNetB, values[0],
      values[1])&&compareNetworks(neuralNetA,neuralNetC[i], values[0], values[1]))
    {
      checkValue = 1;
    }
    freeMemoryForNetwork(neuralNetC[i], values[0], values[1]);
  }
  // create a global mpi variable to store the global score of attackers that managed to
    attack A, B and C
  int result = 0;
  // free the array of neural net c's
  free(neuralNetC);
  // reduce our mpi threads to a single result, if the result is greater than 1 then atleast 1
    thread managed to crack A and B and therefore the attack was successful.
  MPI_Reduce(&checkValue, &result, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
  // if my rank is 0 then check that A and B really were equal, if they were add one to
    values[2] which stores how many times A and B synched, if we successfully attacked as well
      add 1 to values[1].
  if(my_rank==0)
```

```
    {
      if(status==true&&compareNetworks(neuralNetA,neuralNetB, values[0], values[1]))
      {
        values2[0]++;
        if(result>0)
        {
          values2[1]++;
        }
      }
    }
    // free our inputs.
    for (int i = 0; i < values[0]; i++)
    {
      free(inputs[i]);
    }
    free(inputs);
    // free network A and B.
    freeMemoryForNetwork(neuralNetA, values[0], values[1]);
    freeMemoryForNetwork(neuralNetB, values[0], values[1]);
  }
  // if my rank is 0 then print out the statistics.
  if(my_rank==0)
  {
      if(textOff==0) printf("---------- Statistics ----------\nValue of k = %d\nValue of n =
        %d\nValue of l = %d\n# of tests = %d\n# of attackers = %d\nChance of users synchronizing
            = %.2lf\nChance of attackers = %.2lf\nValue of Synchronization = %d\nValue of Epoch =
              %d\n\n",values[0],values[1],values[2],values[3],values[4]*comm_sz,(double)values2[
                0]*100/(double)values2[3],(double)values2[1]*100/(double)values2[
                  0],values[7],values[8]);
      else printf("%d,%d,%d,%d,%d,%.2lf,%.2lf,%d,%d\n",values[
        0],values[1],values[2],values[3],values[4]*comm_sz,(double)values2[0]*100/(double)
          values[3],(double)values2[1]*100/(double)values2[0],values[7],values[8]);
  }
  // finalize mpi.
  MPI_Finalize();
  return 0;
}

// method used to print out networks in the correct order.
void printOutNetworks(struct NeuralNetwork neuralNetA, struct NeuralNetwork neuralNetB, struct
  NeuralNetwork* neuralNetC)
{
  // if my rank is 0 then print out network A and B as well as its own network c's
  if(my_rank == 0)
  {
    printf("Network A\n");
    printNetworkWeights("",neuralNetA,values[0], values[1], values[2], 0);
    printf("\n");
    printf("Network B\n");
    printNetworkWeights("",neuralNetB,values[0], values[1], values[2], 0);
    printf("\n");
    // for each network C print out the network.
    for(int k = 0; k < values[4]; k++)
    {
      char prepend[20];
      prepend[0] = '\0';
      sprintf(prepend, "Network C%d\n", my_rank*values[4]+k+1);
      printNetworkWeights(prepend,neuralNetC[k],values[0], values[1], values[2],0);
      printf("\n");
    }
    // get ready for mpi threads to start sending their networks to be printed. print out in order
      of mpi rank.
    for(int k = 1 ; k < comm_sz; k++)
    {
      // for each mpi thread each thread will create values[4] attackers so wait for each network
        C.
      for(int p = 0; p < values[4]; p++)
      {
        char* printout[values[0]*values[1]*(values[5]+2)+(1*values[0])+50];
        MPI_Recv(&printout, (values[0]*values[1]*(values[5]+2)+(1*values[0])+50), MPI_CHAR, k , 0,
          MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("%s",printout);
        printf("\n");
      }
    }
  }
  else
  {
    // for each mpi thread that is not the host thread send the data of network C to the main
      thread in order of index value.
    for(int k = 0 ; k < values[4]; k++)
    {
      char prepend[20];
      sprintf(prepend, "Network C%d\n", (my_rank*values[4])+k+1);
      char* message = printNetworkWeights(prepend,neuralNetC[k],values[0], values[1], values[2],
        1);
      MPI_Send(message, (values[0]*values[1]*(values[5]+2)+(1*values[0])+50), MPI_CHAR, 0, 0,
        MPI_COMM_WORLD);
      free(message);
    }
  }
}
```

6.2.2 Header File.

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#include <mpi.h>
#include <string.h>
#include <limits.h>
int values[10];
extern int values[11];

// structure to create a neural network
struct NeuralNetwork
{
  int** weights;
  int* hiddenLayerOutputs;
  int networkOutput;
} neuralNet;

// create a definition for booleans
typedef enum
{
  true, false
} bool;

// pointers for methods.
bool runGeometricAttackKKKProtocolParallel(struct NeuralNetwork, struct NeuralNetwork, struct
  NeuralNetwork*, int**, int, int, int, int, int);
bool runKKKProtocolWithoutAttaacker(struct NeuralNetwork, struct NeuralNetwork, int**, int, int,
  int, int, int, int*, int*);
struct NeuralNetwork constructNeuralNetwork(int, int, int);
void initWeights(int**, int, int, int);
void updateWeights(int*, struct NeuralNetwork, int**, int, int, int);
int binaryRand(void);
int** getRandomInputs(int**, int, int);
int getMinInputSumNeuron(struct NeuralNetwork, int**, int, int);
int* getHiddenLayerOutputs(int*, struct NeuralNetwork, int**, int, int);
int getNetworkOutput(int*, struct NeuralNetwork, int**, int, int);
void freeMemoryForNetwork(struct NeuralNetwork, int, int);
char* printNetworkWeights(char*, struct NeuralNetwork, int, int, int, int);
bool compareNetworks(struct NeuralNetwork, struct NeuralNetwork ,int , int );
int numPlaces (int);

bool compareNetworks(struct NeuralNetwork neuralNetA, struct NeuralNetwork neuralNetB ,int k , int
  n)
{
  bool isDifferent = false;
  for(int i = 0 ; i < k; i++)
  {
    for(int j = 0; j < n; j++)
    {
      if(neuralNetA.weights[i][j]!=neuralNetB.weights[i][j])
      {
        isDifferent = true;
      }
    }
  }
  return isDifferent;
}
```

```
/**
 * Simulates the geometric attack on the 3k protocol
 * @param neuralNetA - neuralNetA and neuralNetB are the normal communicating pair by which we
   wish to generate a common key.
 * @param neuralNetB
 * @param attackerNet - or neuralNetC which is for the attacker.
 * @param inputs - the kth 'row' of the 'two-dimensional' array contains the inputs to the kth
   neuron.
 * @param k - identifies the number of hidden neurons.
 * @param n - identifies the n of inputs into each hidden neurons. The total number of inputs to
   the network is therefore N = k*n.
 * @param l - is the bound (-l to l) on the range of values that can be assigned to the weights.
   It is proposed that the bigger the l, the more
 *       difficult it is to break the protocol.
 * @param syncThreshold - if the all the involved networks produce the same weights in
   'syncThreshold' successive rounds,
 *                then we take it that the synchronisation is now stable and we can take the weights
   as final.
 * @param epochLimit  - in case the networks are taking too long to reach synchronisation
   stability, we set this limit on the number of rounds that
 *            can be executed so that we don't run the simulation for ever. This limit will depend
  on the resources available to your simulation
 *            environment.
 * @return true or false indicating whether synchronisation was reached or not. Synchronisation is
  reached when the attack succeeds i.e the attacker succeeds in synchronising its
 *       network weights with that of network A and network B.
 */
bool runGeometricAttackKKKProtocolParallel(struct NeuralNetwork neuralNetA, struct NeuralNetwork
  neuralNetB, struct NeuralNetwork* attackerNet, int** inputs, int k, int n, int l, int
    syncThreshold, int epochLimit)
{
  int s = 0;
  int epoch = 0;
  int* outputC = malloc(sizeof(int)*values[4]);
  // allocate space for hidden layer.
  int** hlOutputsGroupings = malloc(sizeof(int*)*values[4]);
  #pragma omp parallel for num_threads(values[4])
  for(int i = 0 ; i < values[4]; i++)
  {
    hlOutputsGroupings[i] = malloc(sizeof (int) * k);
  }
  while ((s < syncThreshold) && (epoch < epochLimit))
  {
    int outputA = getNetworkOutput(hlOutputsGroupings[0], neuralNetA, inputs, k, n);
    int outputB = getNetworkOutput(hlOutputsGroupings[0], neuralNetB, inputs, k, n);
    if(outputA==outputB)
    {
      //Update the weights of A and B using the anti-Hebbian learning rule.
      updateWeights(hlOutputsGroupings[0], neuralNetA, inputs, k, n, l);
      updateWeights(hlOutputsGroupings[0], neuralNetB, inputs, k, n, l);
      //Increase synchronisation count, s.
      s=s+1;
      #pragma omp parallel for num_threads(values[4])
      for(int i = 0; i < values[4]; i++)
      {
        outputC[i] = getNetworkOutput(hlOutputsGroupings[i], attackerNet[i], inputs, k, n);
        // if network C does not equal then
        if (outputA != outputC[i])
        {
          int kthHidden = getMinInputSumNeuron(attackerNet[i], inputs, k, n);
          //negate the output of the "minimum sum neuron" obtained above.
          attackerNet[i].hiddenLayerOutputs[kthHidden] =
            attackerNet[i].[kthHidden] * (-1);
        }
        //For each C update the weight using the anti-hebbian learning rule
        updateWeights(hlOutputsGroupings[i], attackerNet[i], inputs, k, n, l);
      }
    }
    else
    {
      //Reset the synchronisation count - there was no synchronisation or sychronisation broke
        down in the round.
      s = 0;
    }
    //Get new random inputs for the next round.
    getRandomInputs(inputs, k, n);
    //Increment the round count. We will not run the protocol for ever - we will stop after a
      predefined number of rounds if
    //synchronisation has not been reached by then.
    epoch ++;
  }
  // free output c
  free(outputC);
  // free all hidden output's
  #pragma omp parallel for num_threads(values[4])
  for(int i = 0 ; i < values[4]; i++)
  {
    free(hlOutputsGroupings[i]);
  }
  free(hlOutputsGroupings);
  //Did the above while loop stop because the synchronisation threshold was reached?
  if (s == syncThreshold)
  {
    return true; // We have succesfully synchronised the network. The weights were the same for
      syncThreshold number of rounds!
  }
  return false; //We've exceeded the epoch limit without succeeding in synchronising the network.
}
```

```
/**
 * Simulates the 3k protocol between two networks A and B. After the simulation the network
   weights for both network can be printed to show they are
 * synchronised. Use the utility function printNetworkWeights(...) in this library to print the
   network weights of network A and network B and attacker network.
 *
 * @param neuralNetA - neuralNetA and neuralNetB are the normal communicating pair by which we
   wish to generate a common key.
 * @param neuralNetB
 * @param inputs - the kth 'row' of the 'two-dimensional' array contains the inputs to the kth
   neuron.
 * @param k - identifies the number of hidden neurons.
 * @param n - identifies the n of inputs into each hidden neurons. The total number of inputs to
   the network is therefore N = k*n.
 * @param l - is the bound (-l to l) on the range of values that can be assigned to the weights.
   It is proposed that the bigger the l, the more
 *       difficult it is to break the protocol.
 * @param syncThreshold - if the all the involved networks produce the same weights in
   'syncThreshold' successive rounds,
 *                then we take it that the synchronisation is now stable and we can take the weights
   as final.
 *
 * @param epochLimit - in case the networks are taking too long to reach synchronisation
   stability, we set this limit on the number of rounds that
 *           can be executed so that we don't run the simulation for ever. This limit will depend
   on the resources available to your simulation
 *           environment.
 * @param synchLimit - Max synchronization required.
 * @param epochMax - Max epochs required.
 * @return true or false indicating whether synchronisation was reached or not.
 */
bool runKKKProtocolWithoutAttaacker(struct NeuralNetwork neuralNetA, struct NeuralNetwork
  neuralNetB, int** inputs, int k, int n, int l, int syncThreshold, int epochLimit, int*
    synchLimit, int* epochMax)
{
  int s = 0;
  int epoch = 0;
  int* hlOutputs = malloc(sizeof (int) * k);
  while ((s < syncThreshold) && (epoch < epochLimit))
  {
    int outputA = getNetworkOutput(hlOutputs, neuralNetA, inputs, k, n);
    int outputB = getNetworkOutput(hlOutputs, neuralNetB, inputs, k, n);
    if(outputA==outputB)
    {
      updateWeights(hlOutputs, neuralNetA, inputs, k, n, l);
      updateWeights(hlOutputs, neuralNetB, inputs, k, n, l);
      s=s+1;
    }
    else
    {
      if(s>*synchLimit)
      {
        *synchLimit=s;
      }
      s = 0;
    }
    getRandomInputs(inputs, k, n);
    epoch ++;
  }
  free(hlOutputs);
  if(epoch>*epochMax)
  {
    *epochMax = epoch;
  }
  if (s == syncThreshold)
  {
    return true;
  }
  return false;
}

/**
 * Constructs a new two layered neural network with k perceptrons, n inputs per perceptron and
   weight across each input generated randomly
 * from the range -l to l.
 * @param k
 * @param n
 * @param l
 * @return the newly constructed neural network.
 */
struct NeuralNetwork constructNeuralNetwork(int k, int n, int l)
{
    struct NeuralNetwork neuralNetwork;
    // Allocate memory block for the neural network weights and hiddent layer outputs.
    neuralNetwork.weights = malloc(sizeof (int*) * (k));
    //Allocate memory blocks for the hidden layer outputs.
    neuralNetwork.hiddenLayerOutputs = malloc(sizeof (int) * k);
    for (int i = 0; i < k; i++)
    {
    neuralNetwork.weights[i] = malloc(sizeof (int) * n);
    for (int j = 0; j < n; j++)
    {
      neuralNetwork.weights[i][j] = rand() % (2 * l + 1) - l;
    }
  }
  return neuralNetwork;
}
```

```
/**
 * Gets the neuron/perceptron whose sum of product of inputs and weights is the minimum, of all
   the perceptrons in the network.
 * @param neuralNetwork The network to be processed.
 * @param inputs to the network (not part of the NeuralNetwork structure).
 * @param k The number of perceptrons in the network.
 * @param n  The number of inputs to each perceptron.
 * @return  The index of the minimum input sum neuron.
 */
int getMinInputSumNeuron(struct NeuralNetwork neuralNetwork, int** inputs, int k, int n)
{
  int sum = 0;
  int minSum = 0;
  int minSumNeuron = 0;
  // Calculate the sum of product of inputs and weights for each perceptron, and
  // keep track of the minimum of all the perceptrons.
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < n; j++)
    {
      sum = sum + (inputs[i][j] * neuralNetwork.weights[i][j]);
    }
    //To get absolute value
    sum = abs(sum);
    // If current sum of product of inputs and weights is more than our previous
    // minimum, then we've got a new minimum.
    if ((minSum == 0) || (sum < minSum))
    {
      minSum = sum;
      minSumNeuron = i;
    }
    sum = 0;  // Ready for next perceptron.
  }
  return minSumNeuron;
}

/**
 * Updates the weight vectors of a network using the anti-Hebbian learning rule: w(i) = w(i) -
   output * input(i)
 * @param neuralNet The network whose weight is to be updated.
 * @param inputs The input vector containing the inputs to the network.
 * @param k  The number of perceptrons in the network.
 * @param n  The number of inputs to each perceptron in the network.
 * @param l  The upperbound (l) and lower bound (-l) of weight to be assigned.
 *
 */
void updateWeights(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k, int n, int
  l)
{
  getHiddenLayerOutputs(hlOutputs, neuralNet, inputs, k, n);
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < n; j++)
    {
      // Update the weight using anti-Hebbian learning rule.
      neuralNet.weights[i][j] = neuralNet.weights[i][j] + (hlOutputs[i]*inputs[i][j]);
      if (neuralNet.weights[i][j] < ((-1) * l))
      {
        neuralNet.weights[i][j] = (-1) * l;
      }
      else if (neuralNet.weights[i][j] > l)
      {
        neuralNet.weights[i][j] = l;
      }
    }
  }
}

// get the max number of chars a number can take up.
int numPlaces (int n)
{
  if (n < 0) n = (n == INT_MIN) ? INT_MAX : -n;
  if (n < 10) return 1;
  if (n < 100) return 2;
  if (n < 1000) return 3;
  if (n < 10000) return 4;
  if (n < 100000) return 5;
  if (n < 1000000) return 6;
  if (n < 10000000) return 7;
  if (n < 100000000) return 8;
  if (n < 1000000000) return 9;
  return 10;
}
```

```
// method to create a string that we would like to print / send.
char* printNetworkWeights(char* prepend, struct NeuralNetwork neuralNet, int k, int n, int l, int
  printout)
{
  // store the max space that a network could take up.
  char* str = malloc(sizeof(char)*(k*n*(values[5]+2)+(1*k)+50));
  // create an array to store the max value of a weight.
  char tmp[values[5]+2];
  // empty arrays.
  tmp[0] = '\0';
  str[0] = '\0';
  // add the prepend message to the string.
  strcat(str, prepend);
  // create a string which represents the neural network.
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < n; j++)
    {
      sprintf(tmp, "%d, ", neuralNet.weights[i][j]);
      strcat(str, tmp);
    }
    strcat(str, "\n");
  }
  // if we want to print the string on the thread/node that we are running on then print.
  if(printout==0)
  {
    printf("%s",str);
    free(str);
  }
  return str;
}

/**
 * Generates a random number from the set {-1, 1}.
 * @return The generated random number.
 */
int binaryRand()
{
  int randNum = rand();
  if (randNum % 2 == 0)
  {
    return 1;
  }
  else
  {
    return -1;
  }
}

/**
 * Generates random inputs value (each input value is either -1 or 1), to be used for a neural
 * network with k perceptrons and n inputs per perceptron.
 * @param k
 * @param n
 * @return The input vector generated.
 */
int** getRandomInputs(int** inputs, int k, int n)
{
  //generate and set the inputs.
  for (int i = 0; i < k; i++)
  {
    for (int j = 0; j < n; j++)
    {
      inputs[i][j] = binaryRand();
    }
  }
  return inputs;
}

/**
 * Trigger the hidden layer outputs for the supplied neural network, and then return the hidden
 * layer output vector.
 * @param neuralNet The network whose hidden layer outputs is to be triggered.
 * @param inputs   The inputs to the network.
 * @param k   The number of perceptrons in the network.
 * @param n   The number of inputs to each perceptron.
 * @return    The hidden layer outputs of the supplied network.
 */
int* getHiddenLayerOutputs(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k,
  int n)
{
  for (int i = 0; i < k; i++)
  {
    int sum = 0;
    for (int j = 0; j < n; j++)
    {
      sum = sum - (neuralNet.weights[i][j] * inputs[i][j]);
    }
    //Each hidden layer output must be either -1 or +1. We are interested in
    // only the sign parity(negative or positive) of the output of each perceptron.
    if (sum <= 0)
    {
      hlOutputs[i] = -1;
    }
    else
    {
      hlOutputs[i] = 1;
    }
  }
  return hlOutputs;
}
```

```
/**
 * Trigger the output of the neural network and return it.
 * @param neuralNet The network whose output is to be obtained.
 * @param inputs  The inputs to the network.
 * @param k  The number of perceptrons to the network.
 * @param n The number of inputs to each perceptron in the network.
 * @return The value of the output of the network.
 *
 */
int getNetworkOutput(int* hlOutputs, struct NeuralNetwork neuralNet, int** inputs, int k, int n)
{
  getHiddenLayerOutputs(hlOutputs, neuralNet, inputs, k, n);
  //Obtain the product of all the hidden layer outputs. Since each hidden layer
  //output is either 1 or -1, this product will give us a sign parity (positive or negative).
  int prod = 1;
  for (int i = 0; i < k; i++)
  {
    prod = prod * (hlOutputs[i]);
  }
  return prod;
}

/**
 * Free up the memory allocated for a neural network.
 * @param neuralNet
 * @param k The number of perceptrons in the neural network.
 * @param n  The number of inputs to each perceptron in the network.
 */
void freeMemoryForNetwork(struct NeuralNetwork neuralNet, int k, int n)
{
  // Free memory block for the weight vectors of the neural network;
  for (int i = 0; i < k; i++)
  {
    free(neuralNet.weights[i]);
  }
  free(neuralNet.weights);
  //Free memory for the hidden layer outputs.
  free(neuralNet.hiddenLayerOutputs);
}
```

6.2.3 Results from 1 attacker [K,N,L,Tests Done, amount of attackers, Synchronization chance, Attacker chance, synchronization steps required, epoch's required].

1,3,1,1000,1,96.00,57.92,8,2037
2,3,1,1000,1,87.70,53.25,8,2053
3,3,1,1000,1,88.50,57.18,10,2090
5,3,1,1000,1,86.00,60.70,10,2114
7,3,1,1000,1,88.70,63.70,11,2102
11,3,1,1000,1,88.60,60.84,11,2141
17,3,1,1000,1,83.50,59.64,9,2137
18,3,1,1000,1,85.00,61.76,10,2171
20,3,1,1000,1,89.70,66.11,11,2180
1,4,1,1000,1,97.40,54.83,8,2025
2,4,1,1000,1,99.40,61.97,12,2059
3,4,1,1000,1,98.90,58.65,11,2071
4,4,1,1000,1,95.70,56.64,10,2087
5,4,1,1000,1,98.90,63.70,12,2097
6,4,1,1000,1,97.20,61.11,10,2127
7,4,1,1000,1,97.90,60.16,11,2118
8,4,1,1000,1,98.80,64.37,12,2115
9,4,1,1000,1,99.10,64.98,13,2135
10,4,1,1000,1,99.70,65.30,13,2110
11,4,1,1000,1,97.50,60.72,11,2154
12,4,1,1000,1,96.20,59.25,10,2112
13,4,1,1000,1,100.00,68.70,16,2179
14,4,1,1000,1,95.50,57.80,10,2156
15,4,1,1000,1,98.50,63.45,12,2138
16,4,1,1000,1,99.90,72.17,18,2179
17,4,1,1000,1,97.50,61.54,11,2158
18,4,1,1000,1,99.80,67.84,15,2125
19,4,1,1000,1,96.20,59.36,11,2171
20,4,1,1000,1,96.40,60.17,11,2163
1,5,1,1000,1,96.70,47.88,8,2019
2,5,1,1000,1,99.90,82.68,35,2442
3,5,1,1000,1,99.50,84.42,33,2433
4,5,1,1000,1,99.20,82.46,29,2763
5,5,1,1000,1,99.10,82.44,29,2321
6,5,1,1000,1,98.70,82.47,29,2574
7,5,1,1000,1,97.30,81.81,26,2406
8,5,1,1000,1,99.20,86.79,32,2436
9,5,1,1000,1,99.30,87.01,33,2553
10,5,1,1000,1,99.40,88.43,34,2405
11,5,1,1000,1,99.70,88.97,37,2363
12,5,1,1000,1,99.30,86.91,32,2569
13,5,1,1000,1,99.50,89.45,35,2608
14,5,1,1000,1,98.90,88.68,32,2419
15,5,1,1000,1,98.10,85.52,28,2467
16,5,1,1000,1,98.80,85.93,30,2575
17,5,1,1000,1,99.70,88.67,36,2473
18,5,1,1000,1,99.10,86.38,31,2426
19,5,1,1000,1,98.70,86.83,29,2366
20,5,1,1000,1,98.20,83.81,28,2406
1,6,1,1000,1,96.10,47.97,8,2020
2,6,1,1000,1,99.50,62.81,14,2116
3,6,1,1000,1,95.10,56.36,11,2100
4,6,1,1000,1,98.70,63.93,14,2237
5,6,1,1000,1,99.60,67.77,18,2129
6,6,1,1000,1,98.00,64.90,14,2182
7,6,1,1000,1,93.50,59.68,11,2164
8,6,1,1000,1,99.50,66.73,17,2185
9,6,1,1000,1,99.70,71.51,19,2180
10,6,1,1000,1,99.80,71.64,20,2190
11,6,1,1000,1,98.80,68.32,15,2160
12,6,1,1000,1,95.30,63.38,12,2205
13,6,1,1000,1,99.70,71.51,19,2158
14,6,1,1000,1,98.50,66.80,14,2260
15,6,1,1000,1,99.10,67.71,15,2174
16,6,1,1000,1,97.90,65.47,14,2211
17,6,1,1000,1,98.70,66.26,14,2174
18,6,1,1000,1,99.50,69.25,17,2252
19,6,1,1000,1,98.80,67.71,14,2203
20,6,1,1000,1,98.90,66.23,14,2191
1,7,1,1000,1,96.90,47.37,8,2021
2,7,1,1000,1,99.40,58.65,13,2085
3,7,1,1000,1,96.70,58.22,11,2095
4,7,1,1000,1,99.80,67.94,18,2110
5,7,1,1000,1,99.80,63.83,16,2147
6,7,1,1000,1,98.50,64.87,13,2156
7,7,1,1000,1,99.70,68.81,16,2136
8,7,1,1000,1,98.90,66.03,14,2162
9,7,1,1000,1,97.90,61.18,13,2166
10,7,1,1000,1,98.90,62.08,13,2148
11,7,1,1000,1,97.30,63.51,13,2199
12,7,1,1000,1,99.20,64.42,14,2164
13,7,1,1000,1,99.30,65.96,14,2173
14,7,1,1000,1,97.00,59.48,12,2154
15,7,1,1000,1,99.40,69.11,16,2212
16,7,1,1000,1,99.90,67.47,19,2168
17,7,1,1000,1,97.50,63.59,13,2170
18,7,1,1000,1,99.80,70.04,18,2198
19,7,1,1000,1,99.30,67.17,15,2212
20,7,1,1000,1,94.00,60.74,11,2208
1,8,1,1000,1,95.00,41.16,8,2017
2,8,1,1000,1,99.80,80.96,32,2231
3,8,1,1000,1,99.20,81.55,31,2574
4,8,1,1000,1,99.30,79.96,26,2275
5,8,1,1000,1,98.60,80.53,26,2476
6,8,1,1000,1,99.80,84.47,33,2383
7,8,1,1000,1,99.30,82.88,28,2322
8,8,1,1000,1,98.70,80.45,25,2381
9,8,1,1000,1,99.80,85.47,33,2506
10,8,1,1000,1,99.50,84.12,32,2366
11,8,1,1000,1,98.80,82.29,28,2338
12,8,1,1000,1,99.30,85.50,30,2343
13,8,1,1000,1,99.60,85.14,32,2377
14,8,1,1000,1,96.20,79.52,22,2307
15,8,1,1000,1,96.40,79.25,22,2405
16,8,1,1000,1,99.00,82.32,27,2417
17,8,1,1000,1,95.10,76.97,22,2402
18,8,1,1000,1,98.90,83.11,28,2530
19,8,1,1000,1,99.30,82.98,28,2453
20,8,1,1000,1,98.80,82.79,27,2340
1,9,1,1000,1,99.30,48.24,11,2022
2,9,1,1000,1,99.20,68.65,19,2147
3,9,1,1000,1,99.20,70.87,18,2141
4,9,1,1000,1,99.70,74.22,21,2306
5,9,1,1000,1,97.40,66.22,16,2246
6,9,1,1000,1,99.30,68.28,18,2163
7,9,1,1000,1,99.00,71.82,19,2169
8,9,1,1000,1,96.40,67.01,15,2219
9,9,1,1000,1,98.70,69.91,18,2223
10,9,1,1000,1,99.90,77.78,26,2271
11,9,1,1000,1,96.90,67.91,15,2172
12,9,1,1000,1,98.40,68.19,17,2279
13,9,1,1000,1,99.00,70.30,17,2275
14,9,1,1000,1,91.20,65.13,13,2193
15,9,1,1000,1,98.40,71.24,17,2261
16,9,1,1000,1,99.50,73.97,21,2235
17,9,1,1000,1,99.80,75.95,23,2247
18,9,1,1000,1,99.60,75.70,22,2278
19,9,1,1000,1,100.00,75.60,23,2260
20,9,1,1000,1,97.70,70.83,16,2275
1,10,1,1000,1,98.60,46.75,10,2017
2,10,1,1000,1,99.50,65.33,18,2224
3,10,1,1000,1,99.20,69.66,19,2242
4,10,1,1000,1,99.60,69.58,18,2290
5,10,1,1000,1,99.20,69.56,18,2204
6,10,1,1000,1,99.40,72.94,19,2172
7,10,1,1000,1,98.40,69.11,17,2373
8,10,1,1000,1,99.20,71.17,18,2260
9,10,1,1000,1,98.70,69.40,17,2179
10,10,1,1000,1,100.00,72.60,21,2216
11,10,1,1000,1,99.90,78.28,27,2235
12,10,1,1000,1,99.40,69.72,18,2196
13,10,1,1000,1,92.90,64.37,13,2272
14,10,1,1000,1,97.90,67.82,16,2231
15,10,1,1000,1,99.80,70.54,21,2233
16,10,1,1000,1,98.30,67.55,16,2236
17,10,1,1000,1,95.70,64.68,14,2197
18,10,1,1000,1,99.40,71.03,18,2243
19,10,1,1000,1,96.50,65.91,14,2232
20,10,1,1000,1,99.10,67.41,18,2228
1,11,1,1000,1,95.20,39.08,8,2023
2,11,1,1000,1,99.20,75.40,26,2340
3,11,1,1000,1,98.50,75.63,26,2321
4,11,1,1000,1,99.20,80.34,29,2469
5,11,1,1000,1,98.80,81.58,28,2654
6,11,1,1000,1,97.90,81.72,25,2395
7,11,1,1000,1,97.80,76.48,25,2281
8,11,1,1000,1,99.80,87.98,42,2471
9,11,1,1000,1,99.60,85.44,31,2531
10,11,1,1000,1,99.90,87.59,37,2448
11,11,1,1000,1,98.20,78.51,25,2370
12,11,1,1000,1,99.90,89.39,43,2458
13,11,1,1000,1,99.60,83.03,31,2329
14,11,1,1000,1,99.80,88.48,39,2378
15,11,1,1000,1,99.00,81.01,27,2377
16,11,1,1000,1,98.50,82.64,28,2414
17,11,1,1000,1,99.30,84.09,32,2405
18,11,1,1000,1,98.40,82.72,27,2431
19,11,1,1000,1,97.10,82.29,25,2331
20,11,1,1000,1,98.80,80.36,26,2365
1,12,1,1000,1,97.00,37.84,8,2021
2,12,1,1000,1,98.20,67.92,19,2208
3,12,1,1000,1,99.70,77.03,25,2275
4,12,1,1000,1,96.50,70.88,18,2240
5,12,1,1000,1,98.50,71.57,20,2313
6,12,1,1000,1,97.80,73.72,19,2273
7,12,1,1000,1,99.70,80.04,28,2274
8,12,1,1000,1,97.90,74.46,19,2300
9,12,1,1000,1,98.00,73.67,19,2331
10,12,1,1000,1,98.40,73.27,21,2314
11,12,1,1000,1,99.60,77.21,24,2397
12,12,1,1000,1,99.80,75.25,23,2301
13,12,1,1000,1,98.20,74.34,21,2348
14,12,1,1000,1,99.60,78.41,24,2283
15,12,1,1000,1,99.80,80.86,26,2395
16,12,1,1000,1,99.20,76.51,22,2333
17,12,1,1000,1,98.50,78.68,22,2411
18,12,1,1000,1,99.50,76.68,23,2395
19,12,1,1000,1,99.90,81.48,30,2257
20,12,1,1000,1,99.30,77.84,23,2433
1,13,1,1000,1,98.70,45.19,10,2017
2,13,1,1000,1,98.90,66.13,20,2300
3,13,1,1000,1,99.50,72.66,24,2236
4,13,1,1000,1,99.10,73.46,23,2290
5,13,1,1000,1,99.00,70.71,21,2351
6,13,1,1000,1,97.90,71.71,19,2259
7,13,1,1000,1,99.40,74.25,23,2263
8,13,1,1000,1,98.60,74.04,22,2232
9,13,1,1000,1,99.00,74.55,22,2244
10,13,1,1000,1,98.30,77.21,22,2229
11,13,1,1000,1,98.60,75.86,22,2257
12,13,1,1000,1,99.90,79.08,30,2308
13,13,1,1000,1,99.80,81.96,27,2255
14,13,1,1000,1,99.60,79.22,28,2295
15,13,1,1000,1,97.10,73.53,18,2365
16,13,1,1000,1,99.90,77.98,29,2308
17,13,1,1000,1,99.80,78.46,25,2261
18,13,1,1000,1,96.70,72.18,18,2309
19,13,1,1000,1,100.00,86.10,38,2336
20,13,1,1000,1,96.60,72.57,18,2244
1,14,1,1000,1,93.50,35.72,8,2018
2,14,1,1000,1,99.90,81.18,36,2241
3,14,1,1000,1,99.70,77.03,32,2395
4,14,1,1000,1,99.50,82.61,31,2357
5,14,1,1000,1,95.00,76.95,23,2448
6,14,1,1000,1,99.00,83.64,31,2447
7,14,1,1000,1,99.60,82.93,35,2435
8,14,1,1000,1,98.20,82.18,28,2311
9,14,1,1000,1,99.70,86.26,34,2410
10,14,1,1000,1,99.60,84.74,35,2440
11,14,1,1000,1,98.90,83.01,29,2397
12,14,1,1000,1,98.10,83.79,28,2488
13,14,1,1000,1,98.90,81.40,28,2288
14,14,1,1000,1,99.40,85.41,31,2444
15,14,1,1000,1,99.90,87.19,39,2633
16,14,1,1000,1,99.90,87.49,40,2403
17,14,1,1000,1,98.80,83.50,31,2572
18,14,1,1000,1,99.60,85.14,33,2454
19,14,1,1000,1,98.00,79.59,27,2466
20,14,1,1000,1,99.40,85.01,32,2422
1,15,1,1000,1,97.70,39.82,9,2020
2,15,1,1000,1,98.80,71.86,23,2227
3,15,1,1000,1,95.90,74.56,21,2187
4,15,1,1000,1,98.80,75.20,23,2286
5,15,1,1000,1,98.90,76.95,24,2303
6,15,1,1000,1,99.70,81.95,29,2376
7,15,1,1000,1,100.00,86.10,35,2391
8,15,1,1000,1,99.50,82.31,29,2279
9,15,1,1000,1,97.80,75.66,23,2362
10,15,1,1000,1,99.10,80.93,27,2415
11,15,1,1000,1,98.90,79.98,26,2330
12,15,1,1000,1,99.00,78.59,25,2501
13,15,1,1000,1,98.40,75.61,22,2449
14,15,1,1000,1,99.70,83.05,29,2302
15,15,1,1000,1,99.50,82.51,30,2801
16,15,1,1000,1,99.60,86.24,31,2334
17,15,1,1000,1,99.70,85.86,35,2404
18,15,1,1000,1,100.00,86.80,39,2495
19,15,1,1000,1,99.80,83.37,31,2350
20,15,1,1000,1,98.50,78.07,25,2405
1,16,1,1000,1,97.50,41.74,9,2014
2,16,1,1000,1,99.60,70.98,25,2392
3,16,1,1000,1,99.30,78.65,31,2363
4,16,1,1000,1,97.90,76.81,24,2309
5,16,1,1000,1,99.40,79.07,26,2284
6,16,1,1000,1,99.80,82.06,30,2363
7,16,1,1000,1,99.80,81.06,28,2345
8,16,1,1000,1,97.90,77.02,22,2336
9,16,1,1000,1,99.30,81.17,26,2491
10,16,1,1000,1,100.00,84.60,34,2359
11,16,1,1000,1,99.00,80.00,26,2410
12,16,1,1000,1,99.20,80.34,26,2337
13,16,1,1000,1,99.30,79.66,27,2331
14,16,1,1000,1,98.80,80.26,27,2460
15,16,1,1000,1,99.70,82.85,30,2320
16,16,1,1000,1,99.50,80.50,27,2356
17,16,1,1000,1,99.80,79.06,27,2356
18,16,1,1000,1,99.10,80.63,25,2323
19,16,1,1000,1,99.30,81.07,26,2399
20,16,1,1000,1,98.20,77.19,23,2392
1,17,1,1000,1,88.00,32.84,7,2017
2,17,1,1000,1,99.90,81.48,37,2394
3,17,1,1000,1,99.40,83.30,35,2480
4,17,1,1000,1,98.20,79.43,29,2413
5,17,1,1000,1,99.20,82.36,32,2470
6,17,1,1000,1,99.40,84.31,33,2545
7,17,1,1000,1,98.30,83.32,30,2512
8,17,1,1000,1,99.50,86.83,35,2442
9,17,1,1000,1,99.20,87.40,35,2428
10,17,1,1000,1,98.90,83.01,29,2691
11,17,1,1000,1,99.20,84.27,31,2497
12,17,1,1000,1,98.60,85.50,31,2385
13,17,1,1000,1,98.60,85.50,32,2459
14,17,1,1000,1,98.30,83.32,29,2622
15,17,1,1000,1,99.00,87.68,36,2546
16,17,1,1000,1,99.70,87.36,36,2540
17,17,1,1000,1,99.10,87.49,35,2660
18,17,1,1000,1,99.40,85.31,33,2783
19,17,1,1000,1,94.20,80.25,25,2525
20,17,1,1000,1,100.00,92.50,50,2512
1,18,1,1000,1,93.10,36.95,8,2020
2,18,1,1000,1,98.50,67.92,24,2414
3,18,1,1000,1,98.80,73.79,26,2478
4,18,1,1000,1,99.80,82.57,32,2454
5,18,1,1000,1,99.30,81.87,28,2332
6,18,1,1000,1,99.30,81.37,29,2347
7,18,1,1000,1,99.80,85.37,35,2604
8,18,1,1000,1,99.60,85.74,32,2495
9,18,1,1000,1,99.60,82.93,31,2591
10,18,1,1000,1,99.30,83.79,28,2375
11,18,1,1000,1,97.80,79.14,25,2442
12,18,1,1000,1,99.90,88.99,40,2491
13,18,1,1000,1,96.80,82.23,25,2494
14,18,1,1000,1,99.10,84.26,29,2518
15,18,1,1000,1,99.80,85.27,32,2570
16,18,1,1000,1,99.80,86.47,35,2409
17,18,1,1000,1,98.30,83.93,27,2567
18,18,1,1000,1,96.80,79.03,25,2448
19,18,1,1000,1,97.30,79.45,24,2505
20,18,1,1000,1,96.50,82.59,24,2618
1,19,1,1000,1,96.50,39.17,9,2021
2,19,1,1000,1,99.90,77.58,34,2406
3,19,1,1000,1,99.60,80.82,31,2384

```
4,19,1,1000,1,99.70,81.24,33,2301
5,19,1,1000,1,99.50,84.82,35,2301
6,19,1,1000,1,99.30,84.99,31,2426
7,19,1,1000,1,98.60,80.12,27,2446
8,19,1,1000,1,98.50,81.62,28,2350
9,19,1,1000,1,99.40,84.41,30,2368
10,19,1,1000,1,99.70,82.15,34,2427
11,19,1,1000,1,98.20,79.02,27,2532
12,19,1,1000,1,98.80,83.10,27,2377
13,19,1,1000,1,99.10,85.17,31,2711
14,19,1,1000,1,99.70,82.95,30,2571
15,19,1,1000,1,99.40,84.51,30,2363
16,19,1,1000,1,96.40,81.02,24,2384
17,19,1,1000,1,98.20,82.08,26,2509
18,19,1,1000,1,96.60,80.02,24,2477
19,19,1,1000,1,99.90,85.29,42,2576
20,19,1,1000,1,96.30,80.17,25,2504
1,20,1,1000,1,97.10,41.50,9,2015
2,20,1,1000,1,100.00,82.00,42,2390
3,20,1,1000,1,99.70,85.76,40,2437
4,20,1,1000,1,99.40,85.21,37,2500
5,20,1,1000,1,98.50,85.69,34,2408
6,20,1,1000,1,96.20,83.37,29,2960
7,20,1,1000,1,98.80,85.22,34,2453
8,20,1,1000,1,98.10,84.00,31,2922
9,20,1,1000,1,99.50,87.74,40,2567
10,20,1,1000,1,97.60,83.40,30,2560
11,20,1,1000,1,98.80,84.21,33,2461
12,20,1,1000,1,96.90,85.14,29,2398
13,20,1,1000,1,99.50,86.53,36,2423
14,20,1,1000,1,98.40,86.99,34,2522
15,20,1,1000,1,98.90,86.45,34,2526
16,20,1,1000,1,96.90,83.80,29,2775
17,20,1,1000,1,98.50,85.58,33,2508
18,20,1,1000,1,97.80,88.04,33,2750
19,20,1,1000,1,95.00,85.89,26,2494
20,20,1,1000,1,99.50,88.74,39,2744
1,4,2,1000,1,88.10,17.48,29,2294
2,4,2,1000,1,95.00,24.63,31,4657
3,4,2,1000,1,92.50,32.65,26,4839
4,4,2,1000,1,94.40,33.26,39,5136
5,4,2,1000,1,93.20,37.45,28,5428
6,4,2,1000,1,88.60,38.83,23,5886
7,4,2,1000,1,94.30,40.51,31,5922
8,4,2,1000,1,93.80,40.62,30,6534
9,4,2,1000,1,96.00,42.08,42,6852
10,4,2,1000,1,91.60,40.07,26,6385
11,4,2,1000,1,94.90,40.46,35,6331
12,4,2,1000,1,91.10,41.60,26,6718
13,4,2,1000,1,93.40,42.29,31,7213
14,4,2,1000,1,94.10,36.98,37,7455
15,4,2,1000,1,92.40,41.77,34,7756
16,4,2,1000,1,93.90,40.04,31,6769
17,4,2,1000,1,90.80,42.40,28,9812
18,4,2,1000,1,92.00,39.57,26,6995
19,4,2,1000,1,92.60,45.36,29,6862
20,4,2,1000,1,92.30,41.17,30,7693
1,5,2,1000,1,99.10,12.51,32,2189
2,5,2,1000,1,92.20,13.99,21,3038
3,5,2,1000,1,97.30,21.17,35,4409
4,5,2,1000,1,96.50,26.74,27,5478
5,5,2,1000,1,96.50,27.25,28,5509
6,5,2,1000,1,96.40,31.02,29,6238
7,5,2,1000,1,97.60,33.71,35,5796
8,5,2,1000,1,97.60,30.53,30,6456
9,5,2,1000,1,97.70,34.08,37,6163
10,5,2,1000,1,97.50,32.72,28,6871
11,5,2,1000,1,95.80,34.34,26,6714
12,5,2,1000,1,95.60,34.73,26,7469
13,5,2,1000,1,94.70,32.95,22,7237
14,5,2,1000,1,93.70,37.25,22,7104
15,5,2,1000,1,95.70,33.23,26,6740
16,5,2,1000,1,97.90,33.61,39,7320
17,5,2,1000,1,97.20,32.41,32,6789
18,5,2,1000,1,97.50,33.33,41,8641
19,5,2,1000,1,96.40,35.58,26,7275
20,5,2,1000,1,98.30,32.55,37,7438
1,6,2,1000,1,94.70,8.03,22,2113
2,6,2,1000,1,99.30,10.98,34,3898
3,6,2,1000,1,94.80,18.57,23,4207
4,6,2,1000,1,97.10,22.25,24,5592
5,6,2,1000,1,97.20,28.29,27,5358
6,6,2,1000,1,98.30,28.38,29,5423
7,6,2,1000,1,96.90,29.10,25,5879
8,6,2,1000,1,99.20,31.25,31,5733
9,6,2,1000,1,98.80,30.16,32,5950
10,6,2,1000,1,94.80,27.85,24,6570
11,6,2,1000,1,96.90,29.82,24,8083
12,6,2,1000,1,99.00,31.31,28,7223
13,6,2,1000,1,98.60,31.03,28,8595
14,6,2,1000,1,99.20,32.46,30,7826
15,6,2,1000,1,98.10,30.58,26,8859
16,6,2,1000,1,99.00,31.21,28,8315
17,6,2,1000,1,96.00,31.15,23,9062
18,6,2,1000,1,98.40,33.33,33,8398
19,6,2,1000,1,95.90,29.30,24,8179
20,6,2,1000,1,97.60,31.76,27,8097
1,7,2,1000,1,99.80,10.32,37,2106
2,7,2,1000,1,98.50,9.24,30,2843
3,7,2,1000,1,97.40,13.66,28,3694
4,7,2,1000,1,99.80,20.94,42,6406
5,7,2,1000,1,91.10,24.70,22,5371
6,7,2,1000,1,96.80,25.93,25,6077
7,7,2,1000,1,99.00,26.06,31,6575
8,7,2,1000,1,98.40,27.03,30,7494
9,7,2,1000,1,99.00,27.78,30,6605
10,7,2,1000,1,99.30,28.20,34,6579
11,7,2,1000,1,99.40,30.58,34,6818
12,7,2,1000,1,99.20,29.94,35,6819
13,7,2,1000,1,98.50,30.46,29,6855
14,7,2,1000,1,99.30,30.41,31,6738
15,7,2,1000,1,98.40,27.64,31,8629
16,7,2,1000,1,94.70,29.88,23,7584
17,7,2,1000,1,98.80,30.97,32,7286
18,7,2,1000,1,99.60,28.11,34,7163
19,7,2,1000,1,99.70,29.49,40,8017
20,7,2,1000,1,96.30,31.15,26,9525
1,8,2,1000,1,95.80,8.04,25,2089
2,8,2,1000,1,99.20,11.09,30,2857
3,8,2,1000,1,99.70,13.24,36,4152
4,8,2,1000,1,92.10,18.78,22,5099
5,8,2,1000,1,94.30,23.65,24,5064
6,8,2,1000,1,99.00,23.74,35,5329
7,8,2,1000,1,99.00,26.26,31,6417
8,8,2,1000,1,97.60,26.33,28,7508
9,8,2,1000,1,99.70,28.08,40,6924
10,8,2,1000,1,98.10,30.78,31,6702
11,8,2,1000,1,99.20,25.10,32,6954
12,8,2,1000,1,98.00,27.55,29,7272
13,8,2,1000,1,98.50,30.56,29,7152
14,8,2,1000,1,95.60,28.77,25,7232
15,8,2,1000,1,99.90,30.13,42,8070
16,8,2,1000,1,98.90,27.81,32,8137
17,8,2,1000,1,88.70,29.99,20,6900
18,8,2,1000,1,99.20,30.95,32,7255
19,8,2,1000,1,98.70,29.48,33,7662
20,8,2,1000,1,100.00,28.60,38,7862
1,9,2,1000,1,96.70,8.38,26,2094
2,9,2,1000,1,94.50,9.63,27,2934
3,9,2,1000,1,98.70,16.82,34,3630
4,9,2,1000,1,98.50,21.32,34,4341
5,9,2,1000,1,99.20,22.98,40,5291
6,9,2,1000,1,99.60,27.01,38,5255
7,9,2,1000,1,99.00,26.06,32,5986
8,9,2,1000,1,99.00,27.37,33,6226
9,9,2,1000,1,99.80,30.26,45,7656
10,9,2,1000,1,99.80,26.55,43,6207
11,9,2,1000,1,97.10,28.01,29,6731
12,9,2,1000,1,98.30,28.48,34,8063
13,9,2,1000,1,98.80,27.73,32,8610
14,9,2,1000,1,100.00,27.50,41,7013
15,9,2,1000,1,97.50,27.90,31,7635
16,9,2,1000,1,100.00,29.50,47,7504
17,9,2,1000,1,99.60,30.82,38,7475
18,9,2,1000,1,99.60,27.81,38,7999
19,9,2,1000,1,99.00,27.17,36,7736
20,9,2,1000,1,97.50,28.51,31,8337
1,10,2,1000,1,99.70,10.33,38,2089
2,10,2,1000,1,99.20,11.69,39,3006
3,10,2,1000,1,100.00,16.20,46,3692
4,10,2,1000,1,98.20,20.98,32,4472
5,10,2,1000,1,98.60,21.30,34,4847
6,10,2,1000,1,99.70,23.17,42,5417
7,10,2,1000,1,99.80,25.45,44,5846
8,10,2,1000,1,99.60,28.71,43,7347
9,10,2,1000,1,99.50,29.35,39,6478
10,10,2,1000,1,99.80,27.66,41,6593
11,10,2,1000,1,99.40,30.58,39,6296
12,10,2,1000,1,99.90,29.23,52,6428
13,10,2,1000,1,99.70,27.08,41,6827
14,10,2,1000,1,99.40,28.17,41,7573
15,10,2,1000,1,99.40,27.77,38,7146
16,10,2,1000,1,99.50,29.35,38,8248
17,10,2,1000,1,97.60,26.84,32,7318
18,10,2,1000,1,98.80,29.15,36,7526
19,10,2,1000,1,98.40,29.37,32,8093
20,10,2,1000,1,99.40,30.58,39,7817
1,11,2,1000,1,99.20,9.78,35,2082
2,11,2,1000,1,98.90,15.47,35,2703
3,11,2,1000,1,98.50,15.94,38,3575
4,11,2,1000,1,99.00,19.39,40,4042
5,11,2,1000,1,99.90,23.52,42,6396
6,11,2,1000,1,98.60,24.95,36,5422
7,11,2,1000,1,99.90,27.73,48,5641
8,11,2,1000,1,98.00,25.51,33,6987
9,11,2,1000,1,99.90,29.23,57,6072
10,11,2,1000,1,99.60,29.22,38,7128
11,11,2,1000,1,99.90,29.23,65,6273
12,11,2,1000,1,99.80,28.06,43,8174
13,11,2,1000,1,99.30,27.19,38,7603
14,11,2,1000,1,97.20,30.25,32,7205
15,11,2,1000,1,99.30,26.89,37,7781
16,11,2,1000,1,99.30,30.31,44,7686
17,11,2,1000,1,97.30,30.83,34,7206
18,11,2,1000,1,98.60,30.93,36,7765
19,11,2,1000,1,92.50,25.30,27,7795
20,11,2,1000,1,99.90,29.23,43,7232
1,12,2,1000,1,99.50,9.25,34,2082
2,12,2,1000,1,99.00,18.99,40,2985
3,12,2,1000,1,97.30,16.65,35,3984
4,12,2,1000,1,99.30,21.45,43,5307
5,12,2,1000,1,99.10,24.72,40,4577
6,12,2,1000,1,100.00,23.50,52,5705
7,12,2,1000,1,97.30,25.49,35,5187
8,12,2,1000,1,99.70,26.48,58,5385
9,12,2,1000,1,96.70,26.58,34,5944
10,12,2,1000,1,99.30,28.70,40,6748
11,12,2,1000,1,99.30,26.89,43,6843
12,12,2,1000,1,98.50,28.02,37,6305
13,12,2,1000,1,97.90,28.19,37,6652
14,12,2,1000,1,100.00,29.90,58,6924
15,12,2,1000,1,98.70,27.05,38,6368
16,12,2,1000,1,99.80,31.16,45,6972
17,12,2,1000,1,99.50,30.05,44,7367
18,12,2,1000,1,99.70,27.28,46,7363
19,12,2,1000,1,98.30,31.03,39,6961
20,12,2,1000,1,99.50,31.66,42,7431
1,13,2,1000,1,98.80,8.60,32,2071
2,13,2,1000,1,98.90,19.62,40,4532
3,13,2,1000,1,99.40,19.62,42,4838
4,13,2,1000,1,99.30,20.34,44,4022
5,13,2,1000,1,99.30,23.77,45,5061
6,13,2,1000,1,98.00,26.84,42,5245
7,13,2,1000,1,99.20,24.19,45,5850
8,13,2,1000,1,99.70,26.78,47,5599
9,13,2,1000,1,99.40,29.58,42,6248
10,13,2,1000,1,99.70,29.89,47,6165
11,13,2,1000,1,99.00,28.99,43,6519
12,13,2,1000,1,99.20,29.84,44,7045
13,13,2,1000,1,96.80,29.75,35,6500
14,13,2,1000,1,99.00,30.81,41,6557
15,13,2,1000,1,96.80,30.48,37,7620
16,13,2,1000,1,100.00,29.50,55,6467
17,13,2,1000,1,99.70,28.79,43,6448
18,13,2,1000,1,99.90,32.23,58,6784
19,13,2,1000,1,99.20,28.83,40,7004
20,13,2,1000,1,97.50,30.36,38,7677
1,14,2,1000,1,96.60,7.56,26,2077
2,14,2,1000,1,99.00,21.41,42,4291
3,14,2,1000,1,98.80,22.98,47,4398
4,14,2,1000,1,96.90,22.08,38,4349
5,14,2,1000,1,98.90,24.17,46,4934
6,14,2,1000,1,100.00,27.80,63,5297
7,14,2,1000,1,98.30,26.86,43,5135
8,14,2,1000,1,98.70,26.55,42,7173
9,14,2,1000,1,99.40,28.77,43,5688
10,14,2,1000,1,93.80,26.65,35,5967
11,14,2,1000,1,97.60,30.64,41,6370
12,14,2,1000,1,98.60,30.83,44,7061
13,14,2,1000,1,100.00,30.30,64,7264
14,14,2,1000,1,98.90,28.51,43,7068
15,14,2,1000,1,99.90,31.63,61,7490
16,14,2,1000,1,99.90,28.63,50,6772
17,14,2,1000,1,96.80,32.02,37,6650
18,14,2,1000,1,99.00,30.81,46,7133
19,14,2,1000,1,98.10,33.03,43,7182
20,14,2,1000,1,98.70,32.12,41,9049
1,15,2,1000,1,99.30,10.07,36,2079
2,15,2,1000,1,98.10,20.90,40,4178
3,15,2,1000,1,99.40,21.83,53,3448
4,15,2,1000,1,99.90,23.42,58,4180
5,15,2,1000,1,97.90,27.58,41,4159
6,15,2,1000,1,98.50,26.60,45,5609
7,15,2,1000,1,98.70,29.48,46,4823
8,15,2,1000,1,99.70,31.19,54,6328
9,15,2,1000,1,99.70,30.39,55,7425
10,15,2,1000,1,99.20,28.33,46,6047
11,15,2,1000,1,99.70,31.09,59,6391
12,15,2,1000,1,99.80,32.67,56,6001
13,15,2,1000,1,100.00,33.00,63,6478
14,15,2,1000,1,99.00,31.31,44,6680
15,15,2,1000,1,96.90,29.72,41,7007
16,15,2,1000,1,98.50,30.46,42,7187
17,15,2,1000,1,97.30,31.55,42,7220
18,15,2,1000,1,99.10,29.16,43,6553
19,15,2,1000,1,99.80,28.76,57,7962
20,15,2,1000,1,99.90,31.83,57,6816
1,16,2,1000,1,96.80,8.78,28,2074
2,16,2,1000,1,99.90,24.52,61,3697
3,16,2,1000,1,99.10,23.92,53,4495
4,16,2,1000,1,98.20,24.13,44,3916
5,16,2,1000,1,99.80,27.15,58,4938
6,16,2,1000,1,99.10,29.47,51,5927
7,16,2,1000,1,99.10,29.97,50,5345
8,16,2,1000,1,99.50,29.95,55,5523
9,16,2,1000,1,98.80,29.76,48,5873
10,16,2,1000,1,99.90,30.03,58,7045
11,16,2,1000,1,99.40,33.40,50,6630
12,16,2,1000,1,99.20,31.25,52,6321
13,16,2,1000,1,98.00,31.12,47,6251
14,16,2,1000,1,98.70,32.22,48,6474
15,16,2,1000,1,99.10,29.77,47,6958
16,16,2,1000,1,99.80,34.97,63,6657
17,16,2,1000,1,98.60,31.14,51,6300
18,16,2,1000,1,99.70,32.60,60,8747
19,16,2,1000,1,98.60,31.44,47,7065
20,16,2,1000,1,99.90,34.03,62,6884
1,17,2,1000,1,99.70,8.73,36,2077
2,17,2,1000,1,98.90,26.29,48,3878
3,17,2,1000,1,99.60,23.80,54,3814
4,17,2,1000,1,99.90,25.13,63,5286
5,17,2,1000,1,99.80,27.05,61,4806
6,17,2,1000,1,99.70,30.89,66,7403
7,17,2,1000,1,99.80,29.86,63,4844
8,17,2,1000,1,99.10,32.90,49,6660
9,17,2,1000,1,99.90,29.23,72,5922
10,17,2,1000,1,99.40,29.88,52,6417
11,17,2,1000,1,99.60,31.93,54,5966
12,17,2,1000,1,99.90,34.23,62,6743
13,17,2,1000,1,99.90,33.73,64,6354
14,17,2,1000,1,97.90,31.15,49,5956
15,17,2,1000,1,99.60,33.53,60,6522
16,17,2,1000,1,98.70,31.61,51,6892
17,17,2,1000,1,100.00,36.40,66,6884
18,17,2,1000,1,99.60,33.13,53,6718
19,17,2,1000,1,99.70,32.90,59,7577
20,17,2,1000,1,99.30,33.03,60,6746
1,18,2,1000,1,98.00,6.84,28,2065
2,18,2,1000,1,98.40,24.19,46,3189
3,18,2,1000,1,98.60,24.75,50,5124
4,18,2,1000,1,99.00,24.55,53,4090
5,18,2,1000,1,99.70,29.09,60,4673
6,18,2,1000,1,99.80,31.26,65,5311
7,18,2,1000,1,99.90,32.63,72,5101
8,18,2,1000,1,99.10,30.17,53,5843
9,18,2,1000,1,98.50,31.17,50,6548
10,18,2,1000,1,99.30,31.82,57,5955
11,18,2,1000,1,100.00,35.60,61,7405
12,18,2,1000,1,97.40,31.83,46,6357
13,18,2,1000,1,99.50,31.06,58,5727
```

```
14,18,2,1000,1,100.00,34.30,70,5977
15,18,2,1000,1,99.90,30.63,69,6579
16,18,2,1000,1,99.40,35.31,55,7393
17,18,2,1000,1,97.40,33.47,49,7112
18,18,2,1000,1,99.30,33.94,57,7572
19,18,2,1000,1,99.20,34.88,54,7240
20,18,2,1000,1,98.70,35.97,53,6761
1,19,2,1000,1,97.00,8.04,29,2068
2,19,2,1000,1,99.30,28.10,56,3443
3,19,2,1000,1,97.80,26.99,49,5169
4,19,2,1000,1,99.00,27.78,54,4043
5,19,2,1000,1,99.70,32.90,70,4522
6,19,2,1000,1,99.00,28.69,59,4912
7,19,2,1000,1,99.90,31.23,60,5655
8,19,2,1000,1,99.60,35.44,61,6349
9,19,2,1000,1,98.60,31.44,55,6174
10,19,2,1000,1,99.90,36.04,81,6701
11,19,2,1000,1,98.90,34.07,58,6331
12,19,2,1000,1,97.90,31.26,49,5574
13,19,2,1000,1,97.20,33.85,50,5958
14,19,2,1000,1,99.90,35.04,79,6338
15,19,2,1000,1,100.00,35.10,61,7524
16,19,2,1000,1,98.90,33.06,56,6995
17,19,2,1000,1,95.20,31.41,45,6781
18,19,2,1000,1,99.60,33.73,62,7236
19,19,2,1000,1,99.60,36.35,69,6384
20,19,2,1000,1,99.40,35.61,59,7014
1,20,2,1000,1,97.70,7.16,30,2055
2,20,2,1000,1,99.40,27.77,57,5225
3,20,2,1000,1,99.80,32.77,70,5342
4,20,2,1000,1,99.10,31.18,57,4465
5,20,2,1000,1,99.30,30.82,58,5546
6,20,2,1000,1,100.00,33.50,79,5222
7,20,2,1000,1,99.20,33.97,61,5825
8,20,2,1000,1,98.90,34.28,59,5403
9,20,2,1000,1,96.90,31.99,47,5959
10,20,2,1000,1,99.80,33.17,63,5984
11,20,2,1000,1,98.90,33.47,58,6021
12,20,2,1000,1,99.60,32.13,60,7748
13,20,2,1000,1,99.30,34.54,59,6116
14,20,2,1000,1,100.00,34.50,80,6689
15,20,2,1000,1,99.60,34.74,66,6442
16,20,2,1000,1,99.20,32.26,58,6894
17,20,2,1000,1,99.50,34.17,67,6392
18,20,2,1000,1,99.80,36.97,69,7689
19,20,2,1000,1,99.70,36.51,68,7059
20,20,2,1000,1,99.00,33.74,56,6898
1,5,3,1000,1,96.80,6.92,70,3637
2,5,3,1000,1,95.10,28.50,51,87374
3,5,3,1000,1,94.70,36.01,57,86510
4,5,3,1000,1,93.20,35.62,54,77393
5,5,3,1000,1,94.60,37.95,51,96188
6,5,3,1000,1,91.90,41.35,42,137896
7,5,3,1000,1,96.20,40.12,66,129537
8,5,3,1000,1,95.70,40.65,59,140424
9,5,3,1000,1,95.60,37.97,54,136908
10,5,3,1000,1,95.40,39.41,49,169597
11,5,3,1000,1,96.50,39.48,59,170982
12,5,3,1000,1,95.70,39.92,65,134838
13,5,3,1000,1,93.90,40.89,52,129551
14,5,3,1000,1,94.40,40.68,48,169705
15,5,3,1000,1,95.80,44.36,54,145619
16,5,3,1000,1,95.00,43.16,58,141208
17,5,3,1000,1,92.90,39.50,49,144944
18,5,3,500,1,94.80,44.51,55,162601
19,5,3,500,1,97.00,39.38,55,168799
20,5,3,500,1,96.60,38.72,60,161985
1,6,3,500,1,97.80,0.82,55,2728
2,6,3,500,1,99.00,12.53,59,73426
3,6,3,500,1,98.40,26.63,59,142710
4,6,3,500,1,98.80,28.74,50,176144
5,6,3,500,1,97.80,32.11,51,186142
6,6,3,500,1,97.40,31.01,63,184247
7,6,3,500,1,98.80,28.34,60,199655
8,6,3,500,1,97.00,33.20,58,183017
9,6,3,500,1,98.20,34.01,65,197806
10,6,3,500,1,95.00,35.16,58,197036
11,6,3,500,1,93.60,33.33,53,181603
12,6,3,500,1,95.80,26.72,59,198530
13,6,3,500,1,95.80,35.49,49,198498
14,6,3,500,1,92.00,35.22,61,198264
15,6,3,500,1,93.40,28.05,62,195439
16,6,3,500,1,91.00,33.19,57,199542
17,6,3,500,1,89.20,30.49,52,199362
18,6,3,500,1,87.00,29.43,50,197601
1,7,3,500,1,97.80,1.02,48,2523
2,7,3,500,1,98.20,2.04,53,37201
3,7,3,500,1,97.80,13.70,53,149642
4,7,3,500,1,96.20,23.08,56,197817
5,7,3,500,1,94.20,21.44,53,196831
6,7,3,500,1,90.20,20.40,88,192604
1,8,3,500,1,97.80,0.91,60,2423
2,8,3,500,1,98.20,1.00,54,35201
3,8,3,500,1,97.80,8,52,189642
1,9,3,500,1,99.80,0.80,72,2302
2,9,3,500,1,99.20,0.00,57,33204
3,9,3,500,1,91.80,6.10,47,199351
1,10,3,500,1,99.60,0.20,62,2252
2,10,3,500,1,98.00,0.41,54,13520
3,10,3,500,1,95.20,4.62,52,190380
1,11,3,500,1,99.80,0.20,87,2238
2,11,3,500,1,97.80,0.00,50,14438
3,11,3,500,1,96.80,3.72,71,166770
1,12,3,500,1,99.00,0.20,57,2251
2,12,3,500,1,96.40,0.21,49,11810
3,12,3,500,1,99.00,3.43,64,199593
1,13,3,500,1,97.40,0.21,48,2226
2,13,3,500,1,98.00,0.00,54,12523
3,13,3,500,1,98.40,4.88,60,183576
1,14,3,500,1,99.60,0.00,66,2190
2,14,3,500,1,97.60,0.00,60,8250
3,14,3,500,1,95.40,2.73,56,161255
4,14,3,500,1,86.00,6.05,63,189009
1,15,3,500,1,92.40,0.22,39,2165
2,15,3,500,1,99.60,0.20,74,9713
3,15,3,500,1,97.40,2.46,69,133685
4,15,3,500,1,86.80,7.14,56,190736
1,16,3,500,1,97.00,0.62,49,2147
2,16,3,500,1,99.80,0.60,109,10065
3,16,3,500,1,98.00,2.24,66,125813
4,16,3,500,1,94.20,7.64,62,188639
1,17,3,500,1,99.40,0.20,69,2176
2,17,3,500,1,99.20,0.00,67,10085
3,17,3,500,1,96.20,2.29,78,91187
4,17,3,500,1,93.60,8.12,71,196965
1,18,3,500,1,99.60,0.40,74,2163
2,18,3,500,1,98.00,0.20,68,11489
3,18,3,500,1,99.40,3.02,99,114195
4,18,3,500,1,94.00,8.30,87,184678
1,19,3,500,1,96.80,0.00,51,2156
2,19,3,500,1,98.20,0.00,65,7728
3,19,3,500,1,98.40,1.22,88,87567
4,19,3,500,1,91.40,5.69,64,191174
1,20,3,500,1,97.20,0.41,52,2193
2,20,3,500,1,97.80,0.00,72,10864
3,20,3,500,1,98.20,2.04,95,66152
4,20,3,500,1,97.00,7.84,90,197276
1,5,4,500,1,87.00,11.26,74,138849
1,6,4,500,1,95.20,4.20,80,16155
1,7,4,500,1,94.40,0.00,68,5897
1,8,4,500,1,98.00,0.00,76,3978
1,9,4,500,1,99.20,0.00,86,3444
1,10,4,500,1,94.60,0.00,68,2884
1,11,4,500,1,99.80,0.00,130,2967
1,12,4,500,1,99.60,0.00,111,2778
1,13,4,500,1,98.20,0.00,92,2625
1,14,4,500,1,98.20,0.00,89,2614
1,15,4,500,1,98.60,0.00,90,2521
1,16,4,500,1,99.60,0.00,105,2491
2,16,4,500,1,94.00,0.00,93,197262
1,17,4,500,1,99.60,0.00,118,2473
2,17,4,500,1,93.80,0.00,85,183300
1,18,4,500,1,96.80,0.00,86,2455
2,18,4,500,1,98.80,0.00,104,185458
1,19,4,500,1,98.80,0.00,94,2375
2,19,4,500,1,97.00,0.00,84,143879
1,20,4,500,1,99.40,0.00,105,2434
2,20,4,500,1,98.80,0.00,126,137247
1,7,5,500,1,96.40,0.83,117,96398
1,8,5,500,1,99.40,0.00,127,34665
1,9,5,500,1,95.80,0.00,96,11855
1,10,5,500,1,97.40,0.00,96,8000
1,11,5,500,1,99.40,0.00,136,5608
1,12,5,500,1,100.00,0.00,151,5650
1,13,5,500,1,99.40,0.00,125,3937
1,14,5,500,1,99.60,0.00,125,4172
1,15,5,500,1,99.20,0.00,131,3461
1,16,5,500,1,99.00,0.00,134,3202
1,17,5,500,1,99.80,0.00,141,2983
1,18,5,500,1,99.40,0.00,149,3158
1,19,5,500,1,99.60,0.00,140,2880
1,20,5,500,1,98.20,0.00,126,3011
1,9,6,500,1,98.40,0.00,166,189141
1,10,6,500,1,97.40,0.00,135,63780
1,11,6,500,1,98.40,0.00,156,27932
1,12,6,500,1,99.20,0.00,157,16025
1,13,6,500,1,99.80,0.00,175,10695
1,14,6,500,1,96.20,0.00,139,8645
1,15,6,500,1,98.40,0.00,157,7149
1,16,6,500,1,99.20,0.00,164,5610
1,12,7,500,1,99.60,0.00,220,193883
1,13,7,500,1,97.00,0.00,175,78318
1,14,7,500,1,99.60,0.00,245,40269
1,15,7,500,1,99.40,0.00,227,24700
1,15,8,500,1,99.00,0.00,244,191065
1,16,7,500,1,94.80,0.00,169,18619
1,16,8,500,1,98.80,0.00,265,103086
1,17,7,500,1,99.00,0.00,206,14554
1,17,8,500,1,99.20,0.00,283,66870
1,18,7,500,1,99.20,0.00,197,10439
1,18,8,500,1,97.40,0.00,240,41404
1,19,7,500,1,99.40,0.00,220,9104
1,19,8,500,1,99.20,0.00,287,33035
1,19,9,500,1,96.60,0.00,305,164201
1,20,7,500,1,97.00,0.00,195,8813
1,20,8,500,1,99.80,0.00,335,22571
1,20,9,500,1,96.40,0.00,295,103210
1,19,6,500,1,99.60,0.00,204,4587
1,20,6,500,1,99.20,0.00,191,3890
1,17,6,500,1,99.80,0.00,205,5747
1,18,6,500,1,99.60,0.00,169,4561
```

6.2.4 Results from 20 attackers [K,N,L,Tests Done, amount of attackers, Synchronization chance, Attacker chance, synchronization steps required, epoch's required].

18,5,3,200,20,93.00,87.10,52,188900
19,5,3,200,20,96.50,83.94,54,188272
20,5,3,200,20,96.00,90.62,81,164662
1,6,3,200,20,98.50,35.03,68,2699
2,6,3,200,20,99.50,50.75,52,75749
3,6,3,200,20,95.50,78.53,54,123873
4,6,3,200,20,98.50,79.70,63,148865
5,6,3,200,20,97.00,80.93,59,166063
6,6,3,200,20,94.50,82.01,49,171916
7,6,3,200,20,96.50,82.90,54,169412
8,6,3,200,20,98.00,81.12,60,194355
9,6,3,200,20,98.50,84.26,57,190811
10,6,3,200,20,97.00,81.44,51,199432
11,6,3,200,20,92.50,83.78,50,189829
12,6,3,200,20,92.50,81.62,49,199630
13,6,3,200,20,92.00,84.78,57,196707
14,6,3,200,20,94.00,86.17,77,195877
15,6,3,200,20,94.00,84.57,52,199794
16,6,3,200,20,93.00,87.63,59,196990
17,6,3,200,20,86.50,85.55,48,195297
18,6,3,200,20,87.00,82.76,61,197245
1,7,3,200,20,99.50,11.06,70,2430
2,7,3,200,20,97.50,26.15,59,41412
3,7,3,200,20,97.50,61.03,56,187857
4,7,3,200,20,95.50,61.78,55,181342
5,7,3,200,20,94.50,68.25,64,199915
6,7,3,200,20,85.50,74.85,47,191560
1,8,3,200,20,99.50,8.06,60,2430
2,8,3,200,20,97.50,13.15,60,41412
3,8,3,200,20,97.50,50.03,58,187857
1,9,3,200,20,96.50,6.74,46,2261
2,9,3,200,20,99.50,2.51,77,20035
3,9,3,200,20,98.50,46.19,59,184843
1,10,3,200,20,98.50,10.15,59,2292
2,10,3,200,20,98.50,3.05,49,15232
3,10,3,200,20,94.00,36.70,64,192674
1,11,3,200,20,98.50,5.08,47,2234
2,11,3,200,20,99.00,0.51,59,16706
3,11,3,200,20,91.00,31.32,55,174041
1,12,3,200,20,97.00,3.09,52,2213
2,12,3,200,20,98.50,0.51,59,10101
3,12,3,200,20,97.50,26.67,55,195094
1,13,3,200,20,94.00,5.85,41,2199
2,13,3,200,20,95.50,0.52,53,10463
3,13,3,200,20,97.50,28.21,56,186642
1,14,3,200,20,99.50,4.52,76,2207
2,14,3,200,20,98.00,2.55,53,11976
3,14,3,200,20,97.00,30.93,67,134910
4,14,3,200,20,92.00,47.28,128,191553
1,15,3,200,20,95.50,6.28,51,2151
2,15,3,200,20,98.00,2.04,62,7729
3,15,3,200,20,97.50,24.10,71,136230
4,15,3,200,20,89.50,44.13,48,199520
1,16,3,200,20,99.50,5.53,63,2174
2,16,3,200,20,96.00,2.08,58,6896
3,16,3,200,20,97.50,19.49,69,125140
4,16,3,200,20,92.50,36.76,67,199687
1,17,3,200,20,99.50,7.04,84,2249
2,17,3,200,20,98.50,0.51,66,8344
3,17,3,200,20,98.00,15.31,82,112202
4,17,3,200,20,93.00,32.80,73,172977
1,18,3,200,20,95.00,4.74,49,2181
2,18,3,200,20,99.50,2.51,80,7521
3,18,3,200,20,98.00,15.82,101,197184
4,18,3,200,20,95.50,42.93,104,196811
1,19,3,200,20,95.50,4.19,51,2154
2,19,3,200,20,100.00,1.00,75,8829
3,19,3,200,20,98.00,12.24,71,84312
4,19,3,200,20,93.50,39.57,66,197908
1,20,3,200,20,99.00,7.07,61,2157
2,20,3,200,20,99.00,4.55,91,8461
3,20,3,200,20,96.50,12.44,65,67549
4,20,3,200,20,97.00,38.14,100,193351
1,5,4,200,20,87.50,88.00,99,159247
1,6,4,200,20,94.00,42.55,77,19748
1,7,4,200,20,99.00,4.55,88,5605
1,8,4,200,20,100.00,2.00,89,4161
1,9,4,200,20,98.50,0.00,79,3517
1,10,4,200,20,99.50,0.00,91,2910
1,11,4,200,20,100.00,0.00,115,2747
1,12,4,200,20,100.00,0.00,119,2646
1,13,4,200,20,98.50,0.00,82,2723
1,14,4,200,20,99.50,0.00,107,2608
1,15,4,200,20,99.50,0.00,86,2521
1,16,4,200,20,99.00,0.00,86,2477
2,16,4,200,20,86.50,0.00,77,175625
1,17,4,200,20,98.50,0.00,103,2495
2,17,4,200,20,96.50,0.00,105,187206
1,18,4,200,20,99.00,0.00,94,2469
2,18,4,200,20,94.50,0.00,84,166243
1,19,4,200,20,99.50,0.00,107,2470
2,19,4,200,20,99.00,0.00,86,135804
1,20,4,200,20,99.00,0.00,97,2339
2,20,4,200,20,99.50,0.00,117,134993
1,7,5,200,20,97.00,14.43,121,124801
1,8,5,200,20,99.00,2.02,110,27187
1,9,5,200,20,99.50,0.00,124,12502
1,10,5,200,20,100.00,0.00,158,8386
1,11,5,200,20,99.00,0.00,137,7161
1,12,5,200,20,98.50,0.00,117,4598
1,13,5,200,20,98.50,0.00,128,4100
1,14,5,200,20,99.50,0.00,133,3941
1,15,5,200,20,100.00,0.00,148,3349
1,16,5,200,20,99.50,0.00,146,3271
1,17,5,200,20,99.50,0.00,110,3006
1,18,5,200,20,100.00,0.00,158,2961
1,19,5,200,20,99.50,0.00,131,3051
1,20,5,200,20,100.00,0.00,139,2811
1,9,6,200,20,98.50,0.51,140,190353
1,10,6,200,20,96.00,0.00,140,61335
1,11,6,200,20,98.50,0.00,137,27562
1,12,6,200,20,99.50,0.00,201,15564
1,13,6,200,20,99.00,0.00,158,10439
1,14,6,200,20,100.00,0.00,207,8367
1,15,6,200,20,100.00,0.00,183,6892
1,16,6,200,20,99.50,0.00,186,5225
1,12,7,200,20,99.50,0.00,191,165249
1,13,7,200,20,100.00,0.00,207,75070
1,14,7,200,20,98.00,0.00,215,39894
1,15,7,200,20,97.50,0.00,174,25763
1,15,8,200,20,95.50,0.00,242,182392
1,16,7,200,20,99.50,0.00,256,18881
1,16,8,200,20,99.50,0.00,236,120953
1,17,7,200,20,98.00,0.00,224,12574
1,17,8,200,20,97.00,0.00,232,64621
1,18,7,200,20,98.50,0.00,222,11748
1,18,8,200,20,98.50,0.00,272,42613
1,19,7,200,20,99.00,0.00,206,12126
1,19,8,200,20,94.50,0.00,211,32132
1,19,9,200,20,98.50,0.00,256,179744
1,20,7,200,20,98.00,0.00,197,8055
1,20,8,200,20,98.00,0.00,253,23355
1,20,9,200,20,98.00,0.00,293,107178
1,19,6,200,20,100.00,0.00,182,4594
1,20,6,200,20,96.00,0.00,147,3982
1,17,6,200,20,100.00,0.00,185,5076
1,18,6,200,20,94.50,0.00,142,4436
1,3,1,200,20,96.00,99.48,8,2037
2,3,1,200,20,84.00,98.81,8,2053
3,3,1,200,20,87.00,99.43,10,2090
5,3,1,200,20,91.00,98.35,10,2114
7,3,1,200,20,93.50,98.40,11,2102
11,3,1,200,20,89.00,98.88,11,2141
17,3,1,200,20,84.00,96.43,9,2137
18,3,1,200,20,83.00,95.18,10,2171
20,3,1,200,20,89.00,96.63,11,2180
1,4,1,200,20,98.50,98.98,8,2025
2,4,1,200,20,99.00,99.49,12,2059
3,4,1,200,20,99.50,98.49,11,2071
4,4,1,200,20,98.00,94.90,10,2087
5,4,1,200,20,99.50,98.49,12,2097
6,4,1,200,20,96.50,99.48,10,2127
7,4,1,200,20,98.50,100.00,11,2118
8,4,1,200,20,99.00,96.46,12,2115
9,4,1,200,20,99.00,98.99,13,2135
10,4,1,200,20,97.50,100.00,13,2110
11,4,1,200,20,97.50,97.95,11,2154
12,4,1,200,20,93.00,96.77,10,2112
13,4,1,200,20,100.00,97.00,16,2179
14,4,1,200,20,95.50,95.81,10,2156
15,4,1,200,20,99.00,98.99,12,2138
16,4,1,200,20,99.50,98.99,18,2179
17,4,1,200,20,99.00,95.96,11,2158
18,4,1,200,20,99.50,98.99,15,2125
19,4,1,200,20,97.00,99.48,11,2171
20,4,1,200,20,98.50,95.94,11,2163
1,5,1,200,20,97.00,100.00,8,2019
2,5,1,200,20,100.00,100.00,35,2442
3,5,1,200,20,100.00,100.00,33,2433
4,5,1,200,20,99.00,98.48,29,2763
5,5,1,200,20,97.50,98.97,29,2321
6,5,1,200,20,99.00,99.49,29,2574
7,5,1,200,20,97.00,99.48,26,2406
8,5,1,200,20,99.50,100.00,32,2436
9,5,1,200,20,99.50,100.00,33,2553
10,5,1,200,20,100.00,100.00,34,2405
11,5,1,200,20,100.00,100.00,37,2363
12,5,1,200,20,99.50,99.50,32,2569
13,5,1,200,20,100.00,100.00,35,2608
14,5,1,200,20,97.50,98.97,32,2419
15,5,1,200,20,98.00,100.00,28,2467
16,5,1,200,20,99.00,99.49,30,2575
17,5,1,200,20,99.50,99.50,36,2473
18,5,1,200,20,99.00,98.99,31,2426
19,5,1,200,20,98.50,100.00,29,2366
20,5,1,200,20,98.50,100.00,28,2406
1,6,1,200,20,97.00,98.97,8,2020
2,6,1,200,20,99.50,98.99,14,2116
3,6,1,200,20,95.50,95.81,11,2100
4,6,1,200,20,98.50,98.98,14,2237
5,6,1,200,20,100.00,98.00,18,2129
6,6,1,200,20,99.00,97.47,14,2182
7,6,1,200,20,94.00,98.40,11,2164
8,6,1,200,20,99.00,100.00,17,2185
9,6,1,200,20,100.00,98.50,19,2180
10,6,1,200,20,100.00,99.50,20,2190
11,6,1,200,20,98.00,97.96,15,2160
12,6,1,200,20,95.50,97.91,12,2205
13,6,1,200,20,100.00,99.00,19,2158
14,6,1,200,20,100.00,99.00,14,2260
15,6,1,200,20,99.50,99.50,15,2174
16,6,1,200,20,99.00,98.99,14,2211
17,6,1,200,20,99.00,96.97,14,2174
18,6,1,200,20,99.00,98.48,17,2252
19,6,1,200,20,98.50,97.97,14,2203
20,6,1,200,20,99.00,98.99,14,2191
1,7,1,200,20,94.50,100.00,8,2021
2,7,1,200,20,98.00,97.96,13,2085
3,7,1,200,20,98.50,95.94,11,2095
4,7,1,200,20,100.00,98.50,18,2110
5,7,1,200,20,100.00,99.00,16,2147
6,7,1,200,20,98.00,96.94,13,2156
7,7,1,200,20,99.00,97.47,16,2136
8,7,1,200,20,99.00,97.47,14,2162
9,7,1,200,20,99.00,95.96,13,2166
10,7,1,200,20,98.50,95.94,13,2148
11,7,1,200,20,98.00,94.90,13,2199
12,7,1,200,20,100.00,97.50,14,2164
13,7,1,200,20,100.00,97.50,14,2173
14,7,1,200,20,95.50,96.86,12,2154
15,7,1,200,20,98.50,94.92,16,2212
16,7,1,200,20,100.00,98.50,19,2168
17,7,1,200,20,99.50,98.49,13,2170
18,7,1,200,20,100.00,99.50,18,2198
19,7,1,200,20,99.50,96.98,15,2212
20,7,1,200,20,98.50,98.98,11,2208
1,8,1,200,20,97.50,98.46,8,2017
2,8,1,200,20,100.00,100.00,32,2231
3,8,1,200,20,99.50,99.50,31,2574
4,8,1,200,20,99.50,100.00,26,2275
5,8,1,200,20,99.00,99.49,26,2476
6,8,1,200,20,99.50,99.50,33,2383
7,8,1,200,20,99.50,97.99,28,2322
8,8,1,200,20,98.50,98.98,25,2381
9,8,1,200,20,99.50,100.00,33,2506
10,8,1,200,20,99.50,97.99,32,2366
11,8,1,200,20,99.00,98.48,28,2338
12,8,1,200,20,100.00,100.00,30,2343
13,8,1,200,20,99.50,100.00,32,2377
14,8,1,200,20,96.50,98.96,22,2307
15,8,1,200,20,95.00,99.47,22,2405
16,8,1,200,20,98.50,98.98,27,2417
17,8,1,200,20,96.50,100.00,22,2402
18,8,1,200,20,99.50,99.50,28,2530
19,8,1,200,20,100.00,99.50,28,2453
20,8,1,200,20,98.50,98.98,27,2340
1,9,1,200,20,98.50,99.49,11,2022
2,9,1,200,20,98.50,100.00,19,2147
3,9,1,200,20,98.50,99.49,18,2141
4,9,1,200,20,99.50,97.49,21,2306
5,9,1,200,20,98.50,96.95,16,2246
6,9,1,200,20,99.00,98.99,18,2163
7,9,1,200,20,99.00,97.47,19,2169
8,9,1,200,20,97.00,96.91,15,2219
9,9,1,200,20,99.00,98.48,18,2223
10,9,1,200,20,100.00,100.00,26,2271
11,9,1,200,20,98.00,97.96,15,2172
12,9,1,200,20,98.50,97.46,17,2279
13,9,1,200,20,98.50,98.48,17,2275
14,9,1,200,20,92.50,97.84,13,2193
15,9,1,200,20,99.00,98.99,17,2261
16,9,1,200,20,99.50,98.99,21,2235
17,9,1,200,20,100.00,100.00,23,2247
18,9,1,200,20,100.00,98.50,22,2278
19,9,1,200,20,100.00,100.00,23,2260
20,9,1,200,20,97.50,98.97,16,2275
1,10,1,200,20,99.50,98.99,10,2017
2,10,1,200,20,99.00,98.48,18,2224
3,10,1,200,20,100.00,98.00,19,2242
4,10,1,200,20,99.50,98.99,18,2290
5,10,1,200,20,99.50,97.99,18,2204
6,10,1,200,20,100.00,98.00,19,2172
7,10,1,200,20,99.00,97.47,17,2373
8,10,1,200,20,99.00,97.47,18,2260
9,10,1,200,20,100.00,98.00,17,2179
10,10,1,200,20,99.00,98.99,21,2216
11,10,1,200,20,100.00,99.50,27,2235
12,10,1,200,20,99.50,98.99,18,2196
13,10,1,200,20,91.50,97.81,13,2272
14,10,1,200,20,99.00,98.48,16,2231
15,10,1,200,20,99.50,98.99,21,2233
16,10,1,200,20,99.50,96.98,16,2236
17,10,1,200,20,96.50,98.45,14,2197
18,10,1,200,20,99.50,97.99,18,2243
19,10,1,200,20,95.50,97.38,14,2232
20,10,1,200,20,99.50,100.00,18,2228
1,11,1,200,20,94.50,98.94,8,2023
2,11,1,200,20,99.50,98.49,26,2340
3,11,1,200,20,98.50,99.49,26,2321
4,11,1,200,20,100.00,99.50,29,2469
5,11,1,200,20,98.50,98.98,28,2654
6,11,1,200,20,99.00,99.49,25,2395
7,11,1,200,20,98.50,97.97,25,2281
8,11,1,200,20,100.00,100.00,42,2471
9,11,1,200,20,99.50,100.00,31,2531
10,11,1,200,20,100.00,100.00,37,2448
11,11,1,200,20,99.00,98.99,25,2370
12,11,1,200,20,100.00,100.00,43,2458
13,11,1,200,20,99.50,100.00,31,2329
14,11,1,200,20,100.00,100.00,39,2378
15,11,1,200,20,98.00,98.98,27,2377
16,11,1,200,20,99.00,99.49,28,2414
17,11,1,200,20,99.00,99.49,32,2405
18,11,1,200,20,99.00,97.98,27,2431
19,11,1,200,20,98.00,99.49,25,2331
20,11,1,200,20,98.00,99.49,26,2365
1,12,1,200,20,92.00,98.91,8,2021
2,12,1,200,20,97.00,99.48,19,2208
3,12,1,200,20,100.00,100.00,25,2275
4,12,1,200,20,97.50,97.44,18,2240
5,12,1,200,20,97.00,98.45,20,2313
6,12,1,200,20,98.00,97.96,19,2273

7,12,1,200,20,100.00,100.00,28,2274
8,12,1,200,20,98.50,97.46,19,2300
9,12,1,200,20,98.00,97.96,19,2331
10,12,1,200,20,99.00,97.98,21,2314
11,12,1,200,20,100.00,100.00,24,2397
12,12,1,200,20,99.00,98.99,23,2301
13,12,1,200,20,99.00,98.99,21,2348
14,12,1,200,20,99.00,98.99,24,2283
15,12,1,200,20,99.50,98.99,26,2395
16,12,1,200,20,100.00,99.50,22,2333
17,12,1,200,20,99.00,99.49,22,2411
18,12,1,200,20,99.50,100.00,23,2395
19,12,1,200,20,100.00,100.00,30,2257
20,12,1,200,20,99.50,98.99,23,2433
1,13,1,200,20,99.00,98.99,10,2017
2,13,1,200,20,99.50,98.99,20,2300
3,13,1,200,20,100.00,100.00,24,2236
4,13,1,200,20,100.00,99.50,23,2290
5,13,1,200,20,99.00,99.49,21,2351
6,13,1,200,20,99.00,97.47,19,2259
7,13,1,200,20,98.00,98.98,23,2263
8,13,1,200,20,99.50,99.50,22,2232
9,13,1,200,20,99.50,98.49,22,2244
10,13,1,200,20,97.50,100.00,22,2229
11,13,1,200,20,100.00,99.50,22,2257
12,13,1,200,20,100.00,99.50,30,2308
13,13,1,200,20,100.00,98.00,27,2255
14,13,1,200,20,100.00,99.00,28,2295
15,13,1,200,20,97.50,97.44,18,2365
16,13,1,200,20,100.00,99.00,29,2308
17,13,1,200,20,100.00,99.00,25,2261
18,13,1,200,20,98.00,98.47,18,2309
19,13,1,200,20,100.00,100.00,38,2336
20,13,1,200,20,97.50,97.95,18,2244
1,14,1,200,20,96.50,99.48,8,2018
2,14,1,200,20,100.00,100.00,36,2241
3,14,1,200,20,100.00,99.50,32,2395
4,14,1,200,20,99.50,99.50,31,2357
5,14,1,200,20,94.50,97.35,23,2448
6,14,1,200,20,98.50,98.48,31,2447
7,14,1,200,20,99.50,100.00,35,2435
8,14,1,200,20,98.50,99.49,28,2311
9,14,1,200,20,99.50,100.00,34,2410
10,14,1,200,20,100.00,99.50,35,2440
11,14,1,200,20,98.00,99.49,29,2397
12,14,1,200,20,98.00,98.47,28,2488
13,14,1,200,20,99.00,99.49,28,2288
14,14,1,200,20,99.00,99.49,31,2444
15,14,1,200,20,100.00,100.00,39,2633
16,14,1,200,20,100.00,100.00,40,2403
17,14,1,200,20,99.00,99.49,31,2572
18,14,1,200,20,99.00,98.48,33,2454
19,14,1,200,20,98.00,99.49,27,2466
20,14,1,200,20,98.50,99.49,32,2422
1,15,1,200,20,97.50,99.49,9,2020
2,15,1,200,20,99.50,100.00,23,2227
3,15,1,200,20,97.00,98.97,21,2187
4,15,1,200,20,97.50,97.44,23,2286
5,15,1,200,20,98.50,99.49,24,2303
6,15,1,200,20,99.50,99.50,29,2376
7,15,1,200,20,100.00,99.50,35,2391
8,15,1,200,20,99.50,100.00,29,2279
9,15,1,200,20,99.00,97.47,23,2362
10,15,1,200,20,99.50,99.50,27,2415
11,15,1,200,20,99.50,98.99,26,2330
12,15,1,200,20,99.50,99.50,25,2501
13,15,1,200,20,98.00,98.98,22,2449
14,15,1,200,20,99.50,98.99,29,2302
15,15,1,200,20,100.00,99.00,30,2801
16,15,1,200,20,100.00,100.00,31,2334
17,15,1,200,20,100.00,100.00,35,2404
18,15,1,200,20,100.00,99.50,39,2495
19,15,1,200,20,100.00,99.00,31,2350
20,15,1,200,20,98.00,100.00,25,2405
1,16,1,200,20,97.00,97.94,9,2014
2,16,1,200,20,100.00,99.50,25,2392
3,16,1,200,20,100.00,100.00,31,2363
4,16,1,200,20,98.50,98.98,24,2309
5,16,1,200,20,99.50,98.49,26,2284
6,16,1,200,20,99.50,98.99,30,2363
7,16,1,200,20,99.50,99.50,28,2345
8,16,1,200,20,96.50,97.41,22,2336
9,16,1,200,20,100.00,98.00,26,2491
10,16,1,200,20,100.00,99.00,34,2359
11,16,1,200,20,100.00,98.50,26,2410
12,16,1,200,20,99.50,98.99,26,2337
13,16,1,200,20,99.50,98.99,27,2331
14,16,1,200,20,100.00,99.50,27,2460
15,16,1,200,20,100.00,99.00,30,2320
16,16,1,200,20,99.50,100.00,27,2356
17,16,1,200,20,98.50,98.48,27,2356
18,16,1,200,20,99.50,99.50,25,2323
19,16,1,200,20,99.00,98.99,26,2399
20,16,1,200,20,98.00,98.47,23,2392
1,17,1,200,20,89.50,97.21,7,2017
2,17,1,200,20,99.50,100.00,37,2394
3,17,1,200,20,100.00,99.50,35,2480
4,17,1,200,20,99.00,100.00,29,2413
5,17,1,200,20,99.00,98.99,32,2470
6,17,1,200,20,99.50,100.00,33,2545
7,17,1,200,20,99.00,100.00,30,2512
8,17,1,200,20,99.50,100.00,35,2442
9,17,1,200,20,99.00,100.00,35,2428
10,17,1,200,20,96.00,99.48,29,2691
11,17,1,200,20,98.00,100.00,31,2497
12,17,1,200,20,99.00,99.49,31,2385
13,17,1,200,20,99.00,99.49,32,2459
14,17,1,200,20,98.00,96.94,29,2622
15,17,1,200,20,98.50,100.00,36,2546
16,17,1,200,20,99.50,100.00,36,2540
17,17,1,200,20,100.00,99.50,35,2660
18,17,1,200,20,100.00,100.00,33,2783
19,17,1,200,20,95.00,98.42,25,2525
20,17,1,200,20,100.00,100.00,50,2512
1,18,1,200,20,94.50,98.94,8,2020
2,18,1,200,20,97.50,97.95,24,2414
3,18,1,200,20,99.00,98.99,26,2478
4,18,1,200,20,100.00,99.50,32,2454
5,18,1,200,20,99.00,99.49,28,2332
6,18,1,200,20,99.50,99.50,29,2347
7,18,1,200,20,99.50,100.00,35,2604
8,18,1,200,20,99.50,99.50,32,2495
9,18,1,200,20,99.00,99.49,31,2591
10,18,1,200,20,99.50,99.50,28,2375
11,18,1,200,20,97.50,98.97,25,2442
12,18,1,200,20,100.00,100.00,40,2491
13,18,1,200,20,98.00,100.00,25,2494
14,18,1,200,20,100.00,99.00,29,2518
15,18,1,200,20,99.50,100.00,32,2570
16,18,1,200,20,100.00,99.50,35,2409
17,18,1,200,20,99.50,99.50,27,2567
18,18,1,200,20,99.50,97.99,25,2448
19,18,1,200,20,97.00,99.48,24,2505
20,18,1,200,20,97.00,96.91,24,2618
1,19,1,200,20,96.50,98.96,9,2021
2,19,1,200,20,100.00,100.00,34,2406
3,19,1,200,20,99.50,99.50,31,2384
4,19,1,200,20,100.00,99.50,33,2301
5,19,1,200,20,100.00,100.00,35,2301
6,19,1,200,20,99.00,99.49,31,2426
7,19,1,200,20,98.50,98.98,27,2446
8,19,1,200,20,99.50,98.99,28,2350
9,19,1,200,20,99.50,100.00,30,2368
10,19,1,200,20,99.50,100.00,34,2427
11,19,1,200,20,99.00,98.99,27,2532
12,19,1,200,20,98.00,100.00,27,2377
13,19,1,200,20,100.00,99.50,31,2711
14,19,1,200,20,99.50,98.99,30,2571
15,19,1,200,20,100.00,98.50,30,2363
16,19,1,200,20,97.00,98.97,24,2384
17,19,1,200,20,97.00,99.48,26,2509
18,19,1,200,20,97.50,99.49,24,2477
19,19,1,200,20,100.00,100.00,42,2576
20,19,1,200,20,99.00,99.49,25,2504
1,20,1,200,20,97.50,97.95,9,2015
2,20,1,200,20,100.00,100.00,42,2390
3,20,1,200,20,100.00,100.00,40,2437
4,20,1,200,20,100.00,99.50,37,2500
5,20,1,200,20,99.00,100.00,34,2408
6,20,1,200,20,96.00,99.48,29,2960
7,20,1,200,20,99.00,97.98,34,2453
8,20,1,200,20,99.00,99.49,31,2922
9,20,1,200,20,99.50,100.00,40,2567
10,20,1,200,20,97.50,99.49,30,2560
11,20,1,200,20,98.50,100.00,33,2461
12,20,1,200,20,97.00,100.00,29,2398
13,20,1,200,20,100.00,99.50,36,2423
14,20,1,200,20,99.00,99.49,34,2522
15,20,1,200,20,98.50,99.49,34,2526
16,20,1,200,20,97.50,98.46,29,2775
17,20,1,200,20,98.00,99.49,33,2508
18,20,1,200,20,99.50,99.50,33,2750
19,20,1,200,20,91.00,99.45,26,2494
20,20,1,200,20,98.00,99.49,39,2744
1,4,2,200,20,89.00,91.01,29,2294
2,4,2,200,20,93.00,80.11,31,4657
3,4,2,200,20,89.00,82.02,26,4839
4,4,2,200,20,96.00,85.42,39,5136
5,4,2,200,20,91.50,88.52,28,5428
6,4,2,200,20,88.50,85.31,23,5886
7,4,2,200,20,91.50,90.71,31,5922
8,4,2,200,20,92.00,90.22,30,6534
9,4,2,200,20,97.00,91.24,42,6852
10,4,2,200,20,91.50,87.43,26,6385
11,4,2,200,20,93.50,91.44,35,6331
12,4,2,200,20,90.00,90.56,26,6718
13,4,2,200,20,93.00,84.95,31,7213
14,4,2,200,20,96.00,88.54,37,7455
15,4,2,200,20,94.00,86.70,34,7756
16,4,2,200,20,91.00,84.62,31,6769
17,4,2,200,20,94.00,89.36,28,9812
18,4,2,200,20,94.50,89.42,26,6995
19,4,2,200,20,94.00,89.89,29,6862
20,4,2,200,20,92.00,89.13,30,7693
1,5,2,200,20,98.50,88.32,32,2189
2,5,2,200,20,94.50,62.43,21,3038
3,5,2,200,20,99.00,75.25,35,4409
4,5,2,200,20,95.50,83.25,27,5478
5,5,2,200,20,96.50,83.94,28,5509
6,5,2,200,20,97.50,81.54,29,6238
7,5,2,200,20,97.00,84.54,35,5796
8,5,2,200,20,98.00,81.63,30,6456
9,5,2,200,20,98.50,85.28,37,6163
10,5,2,200,20,99.00,88.38,28,6871
11,5,2,200,20,95.50,78.53,26,6714
12,5,2,200,20,95.50,83.77,26,7469
13,5,2,200,20,94.50,85.19,22,7237
14,5,2,200,20,91.00,86.81,22,7104
15,5,2,200,20,96.00,86.98,26,6740
16,5,2,200,20,99.50,84.92,39,7320
17,5,2,200,20,99.00,86.36,32,6789
18,5,2,200,20,99.00,80.30,41,8641
19,5,2,200,20,94.00,87.77,26,7275
20,5,2,200,20,97.50,85.64,37,7438
1,6,2,200,20,95.00,73.16,22,2113
2,6,2,200,20,99.50,65.33,34,3898
3,6,2,200,20,96.00,70.83,23,4207
4,6,2,200,20,95.00,71.05,24,5592
5,6,2,200,20,98.50,81.22,27,5358
6,6,2,200,20,99.50,81.41,29,5423
7,6,2,200,20,98.00,82.14,25,5879
8,6,2,200,20,99.00,80.30,31,5733
9,6,2,200,20,99.50,78.89,32,5950
10,6,2,200,20,96.50,82.90,24,6570
11,6,2,200,20,94.00,78.72,24,8083
12,6,2,200,20,98.00,76.02,28,7223
13,6,2,200,20,98.00,85.71,28,8595
14,6,2,200,20,99.00,83.33,30,7826
15,6,2,200,20,97.50,85.64,26,8859
16,6,2,200,20,99.50,88.44,28,8315
17,6,2,200,20,93.00,82.80,23,9062
18,6,2,200,20,99.00,83.33,33,8398
19,6,2,200,20,97.00,81.44,24,8179
20,6,2,200,20,97.00,84.02,27,8097
1,7,2,200,20,99.50,74.87,37,2106
2,7,2,200,20,98.50,57.36,30,2843
3,7,2,200,20,97.00,63.40,28,3694
4,7,2,200,20,99.50,75.38,42,6406
5,7,2,200,20,93.00,74.19,22,5371
6,7,2,200,20,95.00,74.74,25,6077
7,7,2,200,20,98.50,73.10,31,6575
8,7,2,200,20,98.50,75.63,30,7494
9,7,2,200,20,97.50,83.08,30,6605
10,7,2,200,20,100.00,80.50,34,6579
11,7,2,200,20,99.50,78.89,34,6818
12,7,2,200,20,100.00,81.50,35,6819
13,7,2,200,20,97.50,82.56,29,6855
14,7,2,200,20,98.00,84.69,31,6738
15,7,2,200,20,99.00,84.85,31,8629
16,7,2,200,20,95.00,83.16,23,7584
17,7,2,200,20,98.50,80.71,32,7286
18,7,2,200,20,98.00,78.57,34,7163
19,7,2,200,20,100.00,82.00,40,8017
20,7,2,200,20,98.00,83.67,26,9525
1,8,2,200,20,95.50,68.59,25,2089
2,8,2,200,20,99.50,59.80,30,2857
3,8,2,200,20,100.00,67.00,36,4152
4,8,2,200,20,90.00,62.22,22,5099
5,8,2,200,20,94.50,73.54,24,5064
6,8,2,200,20,99.50,82.41,35,5329
7,8,2,200,20,99.50,75.38,31,6417
8,8,2,200,20,98.00,80.10,28,7508
9,8,2,200,20,100.00,79.00,40,6924
10,8,2,200,20,98.50,77.66,31,6702
11,8,2,200,20,100.00,84.00,32,6954
12,8,2,200,20,97.50,76.41,29,7272
13,8,2,200,20,99.00,81.31,29,7152
14,8,2,200,20,96.00,80.21,25,7232
15,8,2,200,20,100.00,80.00,42,8070
16,8,2,200,20,99.00,80.30,32,8137
17,8,2,200,20,87.50,82.86,20,6900
18,8,2,200,20,100.00,86.00,32,7255
19,8,2,200,20,98.50,76.65,33,7662
20,8,2,200,20,100.00,83.00,38,7862
1,9,2,200,20,96.50,64.25,26,2094
2,9,2,200,20,95.00,54.74,27,2934
3,9,2,200,20,96.50,64.77,34,3630
4,9,2,200,20,99.50,67.34,34,4341
5,9,2,200,20,99.00,74.75,40,5291
6,9,2,200,20,100.00,75.00,38,5255
7,9,2,200,20,97.50,76.41,32,5986
8,9,2,200,20,99.00,78.79,33,6226
9,9,2,200,20,99.50,78.39,45,7656
10,9,2,200,20,100.00,79.50,43,6207
11,9,2,200,20,97.50,84.62,29,6731
12,9,2,200,20,98.00,81.63,34,8063
13,9,2,200,20,99.00,79.80,32,8610
14,9,2,200,20,99.50,85.93,41,7013
15,9,2,200,20,98.50,79.70,31,7635
16,9,2,200,20,100.00,82.00,47,7504
17,9,2,200,20,99.00,77.78,38,7475
18,9,2,200,20,99.00,83.33,38,7999
19,9,2,200,20,100.00,83.00,36,7736
20,9,2,200,20,97.50,81.54,31,8337
1,10,2,200,20,99.50,81.41,38,2089
2,10,2,200,20,99.50,59.30,39,3006
3,10,2,200,20,100.00,61.50,46,3692
4,10,2,200,20,95.00,68.95,32,4472
5,10,2,200,20,97.50,78.97,34,4847
6,10,2,200,20,99.50,75.38,42,5417
7,10,2,200,20,100.00,73.00,44,5846
8,10,2,200,20,100.00,80.00,43,7347
9,10,2,200,20,99.00,78.79,39,6478
10,10,2,200,20,100.00,81.50,41,6593
11,10,2,200,20,100.00,78.00,39,6296
12,10,2,200,20,100.00,79.00,52,6428
13,10,2,200,20,99.50,77.39,41,6827
14,10,2,200,20,100.00,78.50,41,7573
15,10,2,200,20,99.50,81.41,38,7146
16,10,2,200,20,100.00,84.50,38,8248
17,10,2,200,20,97.50,82.05,32,7318
18,10,2,200,20,98.00,80.10,36,7526
19,10,2,200,20,98.50,82.23,32,8093
20,10,2,200,20,100.00,85.50,39,7817
1,11,2,200,20,98.50,66.50,35,2082
2,11,2,200,20,99.00,57.58,35,2703
3,11,2,200,20,98.50,61.93,38,3575
4,11,2,200,20,99.50,70.85,40,4042
5,11,2,200,20,100.00,72.00,42,6396
6,11,2,200,20,97.00,73.71,36,5422
7,11,2,200,20,100.00,76.00,48,5641
8,11,2,200,20,98.00,75.00,33,6987
9,11,2,200,20,100.00,77.00,57,6072
10,11,2,200,20,98.00,81.63,38,7128
11,11,2,200,20,100.00,82.50,65,6273
12,11,2,200,20,100.00,81.50,43,8174
13,11,2,200,20,99.00,83.33,38,7603
14,11,2,200,20,96.00,85.42,32,7205
15,11,2,200,20,99.50,82.91,37,7781
16,11,2,200,20,100.00,81.50,44,7686

```
17,11,2,200,20,97.00,83.51,34,7206
18,11,2,200,20,98.00,77.55,36,7765
19,11,2,200,20,90.00,77.22,27,7795
20,11,2,200,20,99.50,79.90,43,7232
1,12,2,200,20,99.00,74.24,34,2082
2,12,2,200,20,96.50,52.85,40,2985
3,12,2,200,20,98.00,62.76,35,3984
4,12,2,200,20,99.50,73.37,43,5307
5,12,2,200,20,99.50,74.37,40,4577
6,12,2,200,20,100.00,83.50,52,5705
7,12,2,200,20,95.50,82.72,35,5187
8,12,2,200,20,100.00,83.50,58,5385
9,12,2,200,20,95.00,80.53,34,5944
10,12,2,200,20,99.50,76.38,40,6748
11,12,2,200,20,99.00,79.80,43,6843
12,12,2,200,20,97.50,78.97,37,6305
13,12,2,200,20,98.50,82.23,37,6652
14,12,2,200,20,100.00,81.00,58,6924
15,12,2,200,20,98.50,75.63,38,6368
16,12,2,200,20,99.00,83.84,45,6972
17,12,2,200,20,99.50,82.41,44,7367
18,12,2,200,20,100.00,86.00,46,7363
19,12,2,200,20,99.00,79.80,39,6961
20,12,2,200,20,100.00,81.50,42,7431
1,13,2,200,20,98.50,66.50,32,2071
2,13,2,200,20,99.00,54.55,40,4532
3,13,2,200,20,98.50,71.07,42,4838
4,13,2,200,20,100.00,68.50,44,4022
5,13,2,200,20,98.50,76.65,45,5061
6,13,2,200,20,98.50,73.10,42,5245
7,13,2,200,20,99.00,74.24,45,5850
8,13,2,200,20,99.50,79.40,47,5599
9,13,2,200,20,99.00,78.28,42,6248
10,13,2,200,20,99.50,78.39,47,6165
11,13,2,200,20,98.50,78.17,43,6519
12,13,2,200,20,100.00,86.50,44,7045
13,13,2,200,20,93.50,83.42,35,6500
14,13,2,200,20,99.50,76.88,41,6557
15,13,2,200,20,94.50,81.48,37,7620
16,13,2,200,20,99.50,79.90,55,6467
17,13,2,200,20,99.00,76.77,43,6448
18,13,2,200,20,100.00,78.50,58,6784
19,13,2,200,20,97.50,81.54,40,7004
20,13,2,200,20,99.00,80.30,38,7677
1,14,2,200,20,97.00,72.16,26,2077
2,14,2,200,20,99.00,63.64,42,4291
3,14,2,200,20,99.50,68.84,47,4398
4,14,2,200,20,96.50,69.95,38,4349
5,14,2,200,20,100.00,73.50,46,4934
6,14,2,200,20,100.00,80.50,63,5297
7,14,2,200,20,98.00,79.59,43,5135
8,14,2,200,20,98.00,82.65,42,7173
9,14,2,200,20,99.00,81.82,43,5688
10,14,2,200,20,94.50,81.48,35,5967
11,14,2,200,20,98.00,77.04,41,6370
12,14,2,200,20,99.50,80.90,44,7061
13,14,2,200,20,100.00,83.00,64,7264
14,14,2,200,20,98.50,81.22,43,7068
15,14,2,200,20,100.00,87.00,61,7490
16,14,2,200,20,99.50,85.43,50,6772
17,14,2,200,20,98.00,80.10,37,6650
18,14,2,200,20,100.00,84.00,46,7133
19,14,2,200,20,100.00,75.50,43,7182
20,14,2,200,20,98.50,81.73,41,9049
1,15,2,200,20,99.50,71.86,36,2079
2,15,2,200,20,97.50,61.54,40,4178
3,15,2,200,20,99.50,70.85,53,3448
4,15,2,200,20,100.00,70.50,58,4180
5,15,2,200,20,98.50,70.56,41,4159
6,15,2,200,20,98.50,79.19,45,5609
7,15,2,200,20,97.50,74.87,46,4823
8,15,2,200,20,99.50,82.41,54,6328
9,15,2,200,20,100.00,78.50,55,7425
10,15,2,200,20,99.00,85.35,46,6047
11,15,2,200,20,100.00,82.50,59,6391
12,15,2,200,20,100.00,81.50,56,6001
13,15,2,200,20,100.00,81.50,63,6478
14,15,2,200,20,99.00,81.31,44,6680
15,15,2,200,20,96.50,83.94,41,7007
16,15,2,200,20,98.00,81.63,42,7187
17,15,2,200,20,98.00,80.61,42,7220
18,15,2,200,20,97.50,82.05,43,6553
19,15,2,200,20,99.50,81.91,57,7962
20,15,2,200,20,99.50,82.41,57,6816
1,16,2,200,20,96.00,65.10,28,2074
2,16,2,200,20,100.00,61.00,61,3697
3,16,2,200,20,99.00,69.19,53,4495
4,16,2,200,20,99.00,76.26,44,3916
5,16,2,200,20,99.50,77.39,58,4938
6,16,2,200,20,98.50,79.19,51,5927
7,16,2,200,20,100.00,78.00,50,5345
8,16,2,200,20,100.00,79.50,55,5523
9,16,2,200,20,98.50,85.28,48,5873
10,16,2,200,20,100.00,81.00,58,7045
11,16,2,200,20,99.00,81.31,50,6630
12,16,2,200,20,100.00,86.00,52,6321
13,16,2,200,20,99.00,84.85,47,6251
14,16,2,200,20,99.50,82.91,48,6474
15,16,2,200,20,99.00,82.83,47,6958
16,16,2,200,20,99.50,79.90,63,6657
17,16,2,200,20,98.50,82.74,51,6300
18,16,2,200,20,100.00,86.00,60,8747
19,16,2,200,20,99.00,84.34,47,7065
20,16,2,200,20,100.00,83.00,62,6884
1,17,2,200,20,100.00,69.50,36,2077
2,17,2,200,20,99.00,62.63,48,3878
3,17,2,200,20,99.00,71.21,54,3814
4,17,2,200,20,100.00,73.00,63,5286
5,17,2,200,20,100.00,76.50,61,4806
6,17,2,200,20,100.00,80.50,66,7403
7,17,2,200,20,99.00,82.32,63,4844
8,17,2,200,20,98.50,82.74,49,6660
9,17,2,200,20,100.00,84.50,72,5922
10,17,2,200,20,99.50,80.40,52,6417
11,17,2,200,20,98.50,84.77,54,5966
12,17,2,200,20,99.50,81.91,62,6743
13,17,2,200,20,100.00,81.00,64,6354
14,17,2,200,20,98.00,79.08,49,5956
15,17,2,200,20,99.50,82.41,60,6522
16,17,2,200,20,98.50,83.76,51,6892
17,17,2,200,20,100.00,76.50,66,6884
18,17,2,200,20,99.50,80.40,53,6718
19,17,2,200,20,99.50,83.92,59,7577
20,17,2,200,20,100.00,88.00,60,6746
1,18,2,200,20,97.00,65.98,28,2065
2,18,2,200,20,99.00,61.11,46,3189
3,18,2,200,20,99.00,75.25,50,5124
4,18,2,200,20,99.50,80.40,53,4090
5,18,2,200,20,99.50,75.38,60,4673
6,18,2,200,20,100.00,81.50,65,5311
7,18,2,200,20,99.50,77.89,72,5101
8,18,2,200,20,98.50,81.22,53,5843
9,18,2,200,20,98.50,86.80,50,6548
10,18,2,200,20,99.50,78.39,57,5955
11,18,2,200,20,99.00,81.31,61,7405
12,18,2,200,20,98.00,78.06,46,6357
13,18,2,200,20,100.00,83.50,58,5727
14,18,2,200,20,99.50,88.44,70,5977
15,18,2,200,20,100.00,86.50,69,6579
16,18,2,200,20,99.50,81.91,55,7393
17,18,2,200,20,97.50,80.51,49,7112
18,18,2,200,20,100.00,89.00,57,7572
19,18,2,200,20,99.50,85.43,54,7240
20,18,2,200,20,99.00,83.84,53,6761
1,19,2,200,20,98.00,71.94,29,2068
2,19,2,200,20,99.00,65.66,56,3443
3,19,2,200,20,99.00,72.22,49,5169
4,19,2,200,20,98.50,76.14,54,4043
5,19,2,200,20,100.00,83.00,70,4522
6,19,2,200,20,100.00,80.00,59,4912
7,19,2,200,20,99.00,80.30,60,5655
8,19,2,200,20,98.50,83.25,61,6349
9,19,2,200,20,99.00,83.84,55,6174
10,19,2,200,20,100.00,81.50,81,6701
11,19,2,200,20,99.50,84.92,58,6331
12,19,2,200,20,97.00,80.41,49,5574
13,19,2,200,20,97.50,82.56,50,5958
14,19,2,200,20,100.00,87.00,79,6338
15,19,2,200,20,99.50,82.41,61,7524
16,19,2,200,20,99.00,86.36,56,6995
17,19,2,200,20,96.00,85.42,45,6781
18,19,2,200,20,99.50,84.92,62,7236
19,19,2,200,20,100.00,85.00,69,6384
20,19,2,200,20,99.50,81.41,59,7014
1,20,2,200,20,98.50,70.56,30,2055
2,20,2,200,20,98.50,60.91,57,5225
3,20,2,200,20,100.00,73.50,70,5342
4,20,2,200,20,98.50,82.23,57,4465
5,20,2,200,20,100.00,82.50,58,5546
6,20,2,200,20,99.50,84.42,79,5222
7,20,2,200,20,100.00,85.00,61,5825
8,20,2,200,20,99.00,84.34,59,5403
9,20,2,200,20,97.00,82.99,47,5959
10,20,2,200,20,100.00,87.00,63,5984
11,20,2,200,20,98.50,82.23,58,6021
12,20,2,200,20,100.00,80.50,60,7748
13,20,2,200,20,100.00,85.00,59,6116
14,20,2,200,20,100.00,83.00,80,6689
15,20,2,200,20,99.00,85.35,66,6442
16,20,2,200,20,100.00,83.50,58,6894
17,20,2,200,20,100.00,84.00,67,6392
18,20,2,200,20,100.00,87.50,69,7689
19,20,2,200,20,99.50,80.40,68,7059
20,20,2,200,20,100.00,85.00,56,6898
1,5,3,200,20,98.00,57.14,70,3637
2,5,3,200,20,92.50,75.14,51,87374
3,5,3,200,20,92.50,83.24,57,86510
4,5,3,200,20,94.00,87.23,54,77393
5,5,3,200,20,93.50,87.70,51,96188
6,5,3,200,20,91.00,87.36,42,137896
7,5,3,200,20,96.00,86.98,66,129537
8,5,3,200,20,97.00,85.05,59,140424
9,5,3,200,20,94.50,93.12,54,136908
10,5,3,200,20,94.50,93.12,49,169597
11,5,3,200,20,95.00,88.95,59,170982
12,5,3,200,20,96.50,89.12,65,134838
13,5,3,200,20,95.50,87.96,52,129551
14,5,3,200,20,94.50,89.95,48,169705
15,5,3,200,20,94.00,90.96,54,145619
16,5,3,200,20,94.00,89.89,58,141208
17,5,3,200,20,94.00,86.17,49,144944
```