```
#include<fstream>
#include<iostream>
#include<vector>
#include<math.h>
#include<iomanip>
#include <time.h>
#include <ctime>
#include <random>
#include <cstdlib>
#include <cstring>
#include<algorithm>
#include <sstream>
#include <unistd.h>
#include <string>
#define D cout<<endl<<"DEBUG"<<endl;
long double betrag(long double wert);
class Neuron;
class Net;
class Weight;

long tempargv1;
long tempargv2;

int stillRunning();

using namespace std;
bool allreadyexists=false;

ifstream getInput;
ofstream ausgabe;
ifstream eingabe;
ofstream genwriter;


long double betrag(long double wert){
        if(wert<0){
                wert*=-1;
                return wert;
        }
        return wert;
}



class Weight{
        public:
        long double weight;


};

class Neuron{

        public:

                void WerteWeiterGeben(int i,int a);
                vector<Weight> weights;
                int anzahlNextNeurons;
                vector<long double> tempStorage;
                long double storedValue;
                void FirstFillWeights();
                void calculateSum();

};

class Net{
        public:
```

```
        int anzahlNeuronen;
        int anzahlWeights;
        int anzahlLayers;
        vector<long double> eingangswerte;
        vector<vector<Neuron> > neuralesNetz;
        void ShowWeights();
        void Save();
        void LoadWeights(vector<int> strukturVektor);
        void GetWerteFromFile();
        void PutWerteToNN(vector<int>strukturVektor);
        void FeedForward(vector<int> strukturVektor);
        void showSteps(vector<int> strukturVektor);
        void flush();
};


Net myNet;


void Net::flush(){

        for(int i=0;i<anzahlLayers;i++){
                for(int a=0;a<neuralesNetz[i][a].anzahlNextNeurons;a++){
                        if(i<anzahlLayers){
                        neuralesNetz[i+1][a].tempStorage.clear();
                        }

                }
        }
}
void Net::PutWerteToNN(vector<int>strukturVektor){

        for(int i=0;i<strukturVektor[0];i++){
                myNet.neuralesNetz[0][i].storedValue=myNet.eingangswerte[i];
        }

}

void Net::FeedForward(vector<int> strukturVektor){

        for(int i=0;i<anzahlLayers;i++){
                for(int a=0;a<strukturVektor[i];a++){

                                neuralesNetz[i][a].WerteWeiterGeben(i,a);

                }

        }

}

void Net::showSteps(vector<int> strukturVektor){
        cout<<setprecision(20);
        for(int i=0;i<strukturVektor.size();i++){
                for(int a=0;a<strukturVektor[i];a++){
                        cout<<neuralesNetz[i][a].storedValue<<"        ";

                }
                cout<<endl;
                if(i<strukturVektor.size()-1){
                cout<<"||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||"<<endl;
        }
        }
        }

void Net::GetWerteFromFile(){
getInput.open("./data/"+(to_string(tempargv1)+".map"));
```

```
                if(!getInput.is_open()){
                                cout<<"CANT LOAD '.map' FILE"<<endl;
                }else{


                vector<vector<string> > data;
                while (getInput)
  {

    string s;
    if (!getline( getInput, s )) break;

    istringstream ss( s );
    vector <string> record;

    while (ss)
    {

      string s;
      if (!getline( ss, s, ';' )) break;
      record.push_back( s );

      string::size_type sz;


    const char* a=s.c_str();

    stringstream strValue;
strValue << a;

unsigned int intValue;
strValue >> intValue;


      myNet.eingangswerte.push_back(intValue);

    }

    data.push_back( record );
  }
  if (!getInput.eof())
  {
    cerr << "File Ended on .map file";

  }

                }

                getInput.close();

}

void Net::LoadWeights(vector<int> strukturVektor){
                int count=0;
eingabe.close();

                eingabe.open(".\\data\\"+to_string(tempargv1)+".saveoption");
                vector <string> record;


                if(!eingabe){
                                cout<<endl<<".saveoption-file "<<"./data/"<<to_string(tempargv1)<<".saveoption"<<" not opened"<<endl;
                                cout<<eingabe;
                throw new exception();
        }

        vector<vector<string> > data;
        while (eingabe)
 {
  string s;
  if (!getline( eingabe, s )) break;

  istringstream ss( s );


  while (ss)
  {

   string s;
   if (!getline( ss, s, ';' )) break;

   record.push_back( s );

   count++;
  }

  data.push_back( record );
 }
 if (!eingabe.eof())
 {
  cerr << "File Ended on .saveoption";

 }

if(record.size()==0){
        cout<<endl<<"EITHER THE FILE IS EMPTY OR THERE IS AN CORRUPTION IN THE STRING"<<endl;
}

int anzahlWeights=strukturVektor[0]*strukturVektor[1]+strukturVektor[1]*strukturVektor[2]+strukturVektor[2]*strukturVektor[3];

if(record.size()<anzahlWeights){
        cout<<endl<<"CRITICAL ERROR: WEIGHTFILE NOT MATCHING THE NETWORK"<<endl;
}
if(record.size()>anzahlWeights){
        cout<<endl<<"WARINING: WEIGHTFILE BIGGER THAN INPUT NEURONS"<<endl;
        cout<<endl<<anzahlWeights<<"---"<<record.size()<<endl;
}
        count = 0;

        for(int i=0;i<anzahlLayers;i++){

                for (int a=0;a<strukturVektor[i];a++){

                        for(int b=0;b<myNet.neuralesNetz[i][a].anzahlNextNeurons;b++){

                                        double long templong;

                                        string tempstr=record[count];

                                int     temp=stoi(tempstr); if(temp==0){templong=0.0;
        }else{
                                        templong=1.0/temp;}

                                        myNet.neuralesNetz[i][a].weights.push_back(Weight{templong});

                                        count++;

                        }
                }
        }

}

void Neuron::FirstFillWeights(){
        srand(time(NULL));


for(int i=0;i<anzahlNextNeurons;i++)      {
        long templong=(rand()%1000)-500;
        long double temp=1.0/templong;

        if(temp==0.0){
                 temp=1.0/(rand()%1000);
                 temp++;
        }

        weights.push_back(Weight{temp});

}

}
void Net::ShowWeights(){
        cout<<setprecision(20);
        for(int i=0;i<anzahlLayers;i++){
                for(int a=0;a<neuralesNetz[i].size();a++){
                        for(int b=0;b<neuralesNetz[i][a].anzahlNextNeurons;b++){
                                cout<<neuralesNetz[i][a].weights[b].weight<<"   ";
                        }
                }
        }

}

void Net::Save(){

        for(int i=0;i<anzahlLayers;i++){

                for(int a=0;a<neuralesNetz[i].size();a++){

                        for(int b=0;b<neuralesNetz[i][a].anzahlNextNeurons;b++){

                                        long double tempWeightsLong;

                                        long double tempWeights=neuralesNetz[i][a].weights[b].weight;
                                        if(tempWeights==0.0){             tempWeightsLong=0;
                }else{
                                         tempWeightsLong=1000.0/tempWeights;


                                        tempWeightsLong/=1000.0;
                                        }

                                        ausgabe<<fixed<<setprecision(1)<<tempWeightsLong;

                                ausgabe<<";";
                        }
                }

        }
}

void Neuron::WerteWeiterGeben(int a,int b){

        for(int i=0;i<anzahlNextNeurons;i++){
```

```
    myNet.neuralesNetz[a+1][i].tempStorage.push_back(storedValue*myNet.neuralesNetz[a][b].weights[i].weight);

    }

}

void Neuron::calculateSum(){
    for(int i=0;i<tempStorage.size();i++){
    storedValue+=tempStorage[i];
}
long double temp=1+betrag(storedValue);
storedValue=storedValue/(temp);

}

void ErstelleNetz(vector<int> strukturVektor){

    srand(time(NULL));
myNet.anzahlLayers=strukturVektor.size();

for(int i=0;i<myNet.anzahlLayers;i++){
    myNet.anzahlNeuronen+=strukturVektor[i];

}

for(int i=0;i<myNet.anzahlLayers;i++){
vector<Neuron> layer;
for(int a=0;a<strukturVektor[i];a++)     {
    layer.push_back(Neuron());

}



myNet.neuralesNetz.push_back(layer);
}

for(int i=0;i<myNet.anzahlLayers;i++){


    for(int a=0;a<strukturVektor[i];a++){

        if(i<strukturVektor.size()-1){

    myNet.neuralesNetz[i][a].anzahlNextNeurons=strukturVektor[i+1];

}
}
}

if(!allreadyexists){

for(int i=0;i<myNet.anzahlLayers;i++){
    for(int a=0;a<strukturVektor[i];a++){

        myNet.neuralesNetz[i][a].FirstFillWeights();

    }

}

}

}

void writetofile(vector<int> strukturVektor){
```

```
int currenthighestpos;
long double currenthighest=0;

for(int i=0;i<strukturVektor[myNet.anzahlLayers-1];i++){


        if(myNet.neuralesNetz[myNet.anzahlLayers-1][i].storedValue>currenthighest){
                currenthighest=myNet.neuralesNetz[myNet.anzahlLayers-1][i].storedValue;
                currenthighestpos=i;

        }
        }

switch(currenthighestpos){
        case 0:genwriter<<"U";                          break;
        case 1:genwriter<<"R";                          break;
        case 2:genwriter<<"D";                          break;
        case 3:genwriter<<"L";                          break;
}



}



int stillRunning(){

        return 1;

}



int main(int argc, char* argv[]){




cout<<setprecision(20);

         tempargv1=strtol(argv[1],NULL,10);
        genwriter.open("./data/"+to_string(tempargv1)+".GEN" );
        ausgabe.open("./data/"+to_string(tempargv1)+".weights");
        eingabe.open("./data/"+(to_string(tempargv1)+".saveoption"));



        if(ausgabe.is_open()){
                //cout<<"SUCCESS CREATING FILE"<<endl;
        }
        if(!eingabe.is_open()){
                cout<<endl<<"CRITICAL ERROR:"<<endl<<"ERROR LOADING INPUT FILE"<<endl;


        }
        vector<int> strukturVektor;


        strukturVektor.push_back(12);

        strukturVektor.push_back(8);
        strukturVektor.push_back(4);

         tempargv2=strtol(argv[2],NULL,10);
```

```
		if(tempargv2==1){
				allreadyexists=true;


		}else if(tempargv2==0) allreadyexists=false;
		ErstelleNetz(strukturVektor);

		if(allreadyexists){
		myNet.LoadWeights(strukturVektor);

		}


		myNet.Save();

		ausgabe.close();

myNet.GetWerteFromFile();
myNet.PutWerteToNN(strukturVektor)   ;
myNet.FeedForward(strukturVektor);



for(int i=1;i<=myNet.anzahlLayers;i++){

		for(int a=0;a<strukturVektor[i];a++){

		myNet.neuralesNetz[i][a].calculateSum();

		}
		myNet.FeedForward(strukturVektor);
}

//myNet.showSteps(strukturVektor);

writetofile(strukturVektor);
genwriter.flush();
genwriter.close();

usleep(10000);




}
```