```
#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>
using namespace std;
ifstream weight;
int tempargv1;
vector<int > data;
vector<int> allRets;
vector<int> best;
vector<int> bestpos;

vector<vector<int> > newGen;
vector<vector<int> > weights;
int tempargv;


ofstream LOG,retsLOG;
string  str1="./data/LOG.LOG";
string  rets="./data/execlRets.LOG";

int findeHighestPos(vector<int>Rets){

int currentHigh,currentHighpos;
currentHigh=Rets[0];
currentHighpos=0;
for(int i=0;i<Rets.size();i++){
        if(Rets[i]>currentHigh){
                currentHigh=Rets[i];
                currentHighpos=i;
        }
}

return currentHighpos;

}


void findWorst(){



        for(int i=0;i<100;i++){
                for(int a=0;a<100;a++){
                if(allRets[a]==i){
                        retsLOG<<i<<endl;
                        return;
                }
                }
        }
        retsLOG<<"the hell?";
}

int findBest(){
        int count=0;
```

```
		while(true){
		ifstream file;
			string st1="./data/";
				st1+=to_string(count);
				st1+=".return";



			if(!ifstream(st1)){

				break;
			}



		file.open(st1);


		int temp;
		file>>temp;
		if(temp>=100){
			cout<<"wrong";
			LOG<<"may be false";
		}
		file>>temp;
		allRets.push_back(temp);

		file.close();
		count++;

}

LOG.open(str1,fstream::app);


for(int i=0;i<4;i++){




		int temp=findeHighestPos(allRets);
int		result=allRets[temp];
		bestpos.push_back(temp);
		/*
		result=max_element(allRets.begin(),allRets.end());
		//LOG<<allRets[distance(allRets.begin(),result)]<<" at "<<distance(allRets.begin(),result)<<endl;

//		cout<<allRets[distance(allRets.begin(),result)]<<" at "<<distance(allRets.begin(),result)<<endl;
		LOG<<allRets[distance(allRets.begin(),result)]<<" at "<<distance(allRets.begin(),result)<<endl;
		bestpos.push_back(distance(allRets.begin(),result));

		*/

		if(i==0){
			retsLOG<<allRets[temp]<<",";
			string befehl="copy ";
			befehl+=".\\data\\";
			befehl+=to_string(temp);
			befehl+=".weights ";
			befehl+=".\\best\\";
			befehl+=to_string(tempargv);
			befehl+=".best";
			cout<<befehl;

			system(befehl.c_str());
```

```
		}
		vector<int>::iterator nth = allRets.begin() + temp;
		allRets.erase(nth);
}

findWorst();
LOG.close();



}



void writesave(){
		srand(time(NULL));
		ofstream save;
		ofstream opt;


		LOG.open(str1,ofstream::app);


		for(int b=0;b<10;b++){

		for(int i=0;i<10;i++){
			string  str1="./data/";
			string  str2="./data/";
			int temp=b*10+i;
			str1+=to_string(temp);
			str2+=to_string(temp);
			str1+=".weights";
			str2+=".saveoption";

		save.open(str1);
		opt.open(str2);

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


			if(rand()%50==0){
					//	LOG<<"Mutation occured at "<<b*10+i<<" from "<<newGen[i][a]<<" to ";
					int temp=rand()%2000-1000;
					//	LOG<<temp<<endl;
					save<<temp<<";";
					opt<<temp<<";";
				}else{

			int temprand=rand()%4;
			save<<weights[temprand][a]<<";";
			opt<<weights[temprand][a]<<";";
			/*
			save<<newGen[i][a]<<";";
			opt<<newGen[i][a]<<";";
			*/
		}
		}

		save.close();
		opt.close();
}
}
LOG.close();
}


void loadWeight(int arg,int pos){
```

```
            vector<int > data;



              string::size_type sz;



            weight.open("./data/"+(to_string(arg)+".weights"));

            if(!weight.is_open()){
                        cout<<"CANT LOAD '.weights' FILE"<<endl;
            }else{


            while (weight)
 {

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

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

   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;




   }

for (int i=0; i< record.size(); i++)
   {
    int num = atoi(record.at(i).c_str());

    data.push_back(num);

   }




 }
 if (!weight.eof())
 {
   cerr << "File Ended on .map file";
```

```
 }
		weights.push_back(data);



}
		weight.close();

}

void showoff(int pos){
		cout<<endl;

		for(int i=0;i<weights[pos].size();i++){

			cout<<weights[pos][i]<<endl;

		}

}


void crossover(){

		srand(time(NULL));

		for(int i=0;i<10;i++){
			vector<int>temp;
			for(int a=0;a<weights[0].size();a++){



								temp.push_back(weights[rand()%4][a]);



			}
			newGen.push_back(temp);
		}



}


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

tempargv=strtol(argv[1],NULL,10);


retsLOG.open(rets,ios_base::app);




		findBest();

/* bestpos.push_back(rand()%100);
		bestpos.push_back(rand()%100);*/

		sleep(1);
		for (int i=0;i<4;i++){
```

```
                data.clear();
                loadWeight(bestpos[i],i);

        }



        crossover();

        writesave();


        LOG.close();


}
```