Task 1

Source code:

```
#include "stdafx.h"
#include "time.h"
#include <iostream>
#include <fstream>

using namespace std;

double generation[100][30];
int inputs[32][5];
int fitness[100];
int answers[32];

ofstream file_best;
ofstream file_chromo;
int gen=0;
void inp()
{
	for(int i=1;i<32;i=i+2)
		inputs[i][4]=1;
	for(int i=2;i<32;i=i+4)
	{
		inputs[i][3]=1;
		inputs[i+1][3]=1;
	}
	for(int i=4;i<32;i=i+8)
	{
		inputs[i][2]=1;
		inputs[i+1][2]=1;
		inputs[i+2][2]=1;
		inputs[i+3][2]=1;
	}
	for(int i=8;i<32;i=i+16)
	{
		inputs[i][1]=1;
		inputs[i+1][1]=1;
		inputs[i+2][1]=1;
		inputs[i+3][1]=1;
		inputs[i+4][1]=1;
		inputs[i+5][1]=1;
		inputs[i+6][1]=1;
		inputs[i+7][1]=1;
	}
	for(int i=16;i<32;i++)
		inputs[i][0]=1;
	for(int i=0;i<32;i++)
		for(int j=0;j<5;j++)
			if(inputs[i][j]==0)
```

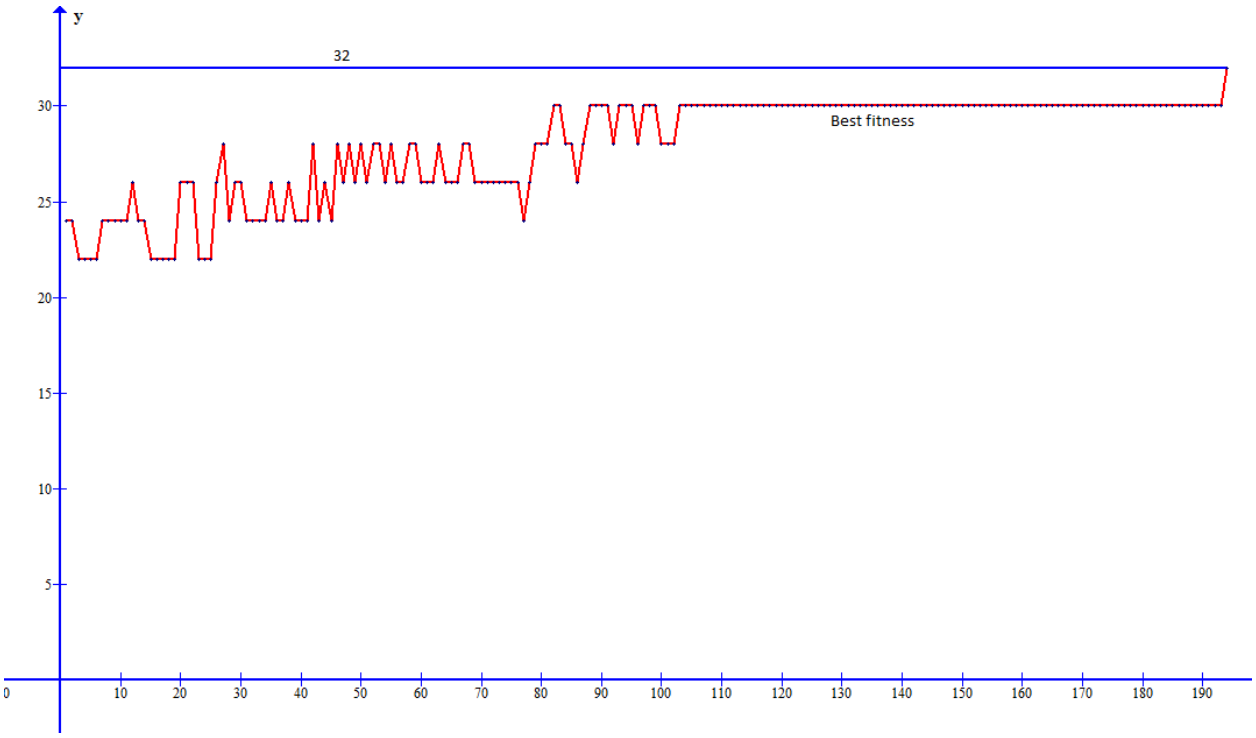
```
                                inputs[i][j]=-1;
}
void answer()
{
        for(int i=0;i<32;i++)
        {
                int ones=0;
                for(int j=0;j<5; j++)
                {
                        if(inputs[i][j]==1)
                                ones++;
                }
                answers[i]=ones%2==0?1:-1;
        }
}
void work()
{
        memset(fitness,0,100*sizeof(int));
        for(int g=0;g<100;g++)
        {
                double neuro[5];
                double output=0;

                for(int i=0;i<32;i++)
                {
                        memset(neuro,0,5*sizeof(double));
                        for(int k=0;k<5;k++)
                        {
                                for(int j=0;j<5;j++)
                                        neuro[k]+=generation[g][k*5+j]*inputs[i][j];
                                if(neuro[k]>=0)
                                        neuro[k]=1;
                                else
                                        neuro[k]=-1;

                        }
                        output=0;
                        for(int k=0;k<5;k++)
                        {
                                output+=neuro[k]*generation[g][25+k];
                        }
                        if(output>=0)
                                output=1;
                        else
                                output=-1;
                        if(output==answers[i])
                                fitness[g]++;
                }
        }
}
```

```
void sort()
{
	for(int i=0;i<100;i++)
		for(int j=0;j<100;j++)
			if(i!=j &&((i<j&&fitness[i]<fitness[j])||(i>j&&fitness[i]>fitness[j])))
			{
				for(int k=0;k<30;k++)
				{
					double gen_t=generation[i][k];
					generation[i][k]=generation[j][k];
					generation[j][k]=gen_t;
				}
				int fit_t = fitness[i];
				fitness[i]=fitness[j];
				fitness[j]=fit_t;
			}
}
void mutation()
{
	for(int i=0;i<100;i++)
		for(int j=0;j<30;j++)
			if(rand()%30==0)
			{
				generation[i][j]=(double)(rand()%1001)/1000;
				if(rand()%2)
					generation[i][j]=-generation[i][j];
			}
}
void cross()
{
	double new_gener[100][30];
	for(int i=0;i<100;i++)
	{
		int mama=rand()%50;
		int papa=rand()%50;
		for(int j=0;j<30;j++)
			if(rand()%2==0)
				new_gener[i][j]=generation[mama][j];
			else
				new_gener[i][j]=generation[papa][j];
	}
	memcpy(generation,new_gener,100*30*sizeof(double));
}
void stats()
{
	file_best<<fitness[0]<<endl;
	file_chromo<<++gen<<endl<<"_____________"<<endl;
	for(int i=0;i<30;i++)
		file_chromo<<generation[0][i]<<endl;
```

```
}
int _tmain(int argc, _TCHAR* argv[])
{
	srand(time(NULL));
	for(int i=0;i<100;i++)
	{
		for(int j=0;j<30;j++)
		{
			generation[i][j]=(double)(rand()%1001)/1000;
			if(rand()%2)
				generation[i][j]=-generation[i][j];
		}
	}
	file_best.open("best.txt",ios_base::trunc);
	file_chromo.open("chromo.txt",ios_base::trunc);
	inp();
	answer();
	for(int i=0;i<5000;i++)
	{
		work();
		sort();
		cross();
		mutation();
		cout<<fitness[0]<<endl;
		stats();
		if(fitness[0]==32)
		{
			cout<<"END"<<endl;
			file_best.close();
			file_chromo.close();
			exit(0);
		}
	}
	return 0;
}
```



Generation 1;

Wij	Etalon	Result
-0.219	1	-1
0.074	-1	1
-0.541	-1	-1
0.91	1	1
-0.641	-1	1
0.317	1	1
0.494	1	-1
0.121	-1	1
0.304	-1	-1
0.084	1	1
0.687	1	-1
0.087	-1	-1
0.806	1	1
0.025	-1	1
0.311	-1	-1
-0.181	1	-1
-0.114	-1	1
-0.621	1	1
0.622	1	-1
0.005	-1	-1
-0.142	1	1
-0.013	-1	1
0.009	-1	-1
0.664	1	1
-0.814	1	-1
-0.538	-1	1

-0.199	-1	-1
0.054	1	-1
0.366	-1	-1
-0.072	1	1
	1	-1
	-1	1

140th generation

Wij	Etalon	Result
0.593	1	1
0.965	-1	-1
0.771	-1	-1
0.929	1	1
0.77	-1	-1
-0.602	1	1
0.546	1	1
-0.908	-1	-1
-0.558	-1	1
-0.53	1	1
0.213	1	1
-0.909	-1	-1
0.436	1	1
0.266	-1	-1
0.399	-1	-1
-0.025	1	1
-0.035	-1	-1
0.87	1	1
0.307	1	1
-0.57	-1	-1
-0.455	1	1
-0.276	-1	-1
0.385	-1	-1
-0.631	1	-1
-0.043	1	1
-0.997	-1	-1
-0.756	-1	-1
-0.854	1	1
-0.116	-1	-1
-0.13	1	1
	1	1
	-1	-1

Last generation

Wij	Etalon	Result
0.609	1	1

0.7	-1	-1
0.771	-1	-1
0.641	1	1
0.649	-1	-1
-0.602	1	1
0.528	1	1
-0.625	-1	-1
-0.56	-1	-1
-0.602	1	1
0.213	1	1
-0.902	-1	-1
0.336	1	1
0.266	-1	-1
0.232	-1	-1
-0.287	1	1
0.873	-1	-1
0.943	1	1
0.594	1	1
-0.57	-1	-1
0.044	1	1
-0.027	-1	-1
-0.294	-1	-1
-0.689	1	1
-0.538	1	1
-0.978	-1	-1
-0.756	-1	-1
-0.854	1	1
0.096	-1	-1
-0.13	1	1
	1	1
	-1	-1