Kiryl Kiryluk

1. Maximize the following y in 3-D cases.

$$y= x_1*\sin(|x_1|)+ x_2*\sin(|x_2|)$$

2. Create a population of 20 chromosomes at random, with fitness being y.
3. Evole this population till fitness dosen't change

CODE:

```
#include <stdafx.h>
#include <fstream>
#include <iostream>
#include <vector>
#include <algorithm>
#include <time.h>
#include <math.h>

typedefvector<Chromosome> Population;
int Chromosome::getGen(int index)
{
	return genes[index];
}
boolsortPopulation(Chromosome &chromosome1, Chromosome &chromosome2);
Population distance(Population population);

voidsetGen(int index, intnewGen);
	void mutation();

voidcurrentPosition();
	intgetDistance();
	intgetGen(int index);


void Chromosome::currentPosition()
{
	for (int i = 0; i <genes.size(); i++)
	{
		switch (genes[i])
{
	return distance;
}


Population uniformDistance(Population population);
int XY(inttmp);
void Chromosome::setGen(int index, intnewGen)
{
	genes[index] = newGen;
}

void Chromosome::mutation()
{
	for (int i = 0; i <genes.size(); i++)
	{
		inttmp = rand() % 100;
```

```
			if (tmp == 0)
			{
				genes[i] = rand() % 4 + 1;
			}
		}
}


boolsortPopulation(Chromosome &chromosome1, Chromosome &chromosome2)
{
		return chromosome1.getDistance() < chromosome2.getDistance();
}

Population distance(Population population)
{
		intdistancePoint;
		intfirstParent, secondParent;
		Population result;

		for (int i = 0; i < 10; i++)
		{
			distancePoint = rand() % 1000;
			firstParent = rand() % 10;
			secondParent = rand() % 10;
			while (firstParent != secondParent)
			{
				secondParent = rand() % 10;
			}

Population uniformDistance(Population population)
{
		intfirstParent, secondParent;
		Population result;

		for (int i = 0; i < 10; i++)
		{
			firstParent = rand() % 10;
			secondParent = rand() % 10;
			while (firstParent != secondParent)
			{
				secondParent = rand() % 10;
			}


intmain()
{
		srand(time(NULL));
		ofstream file("OutData.txt", ios::out);
		ofstreamfileTable("Table.txt", ios::out);

		Population population;
		Population table;
		for (int i = 0; i < 20; i++)
		{
			Chromosome tmp;
			population.push_back(tmp);
		}
		for (int i = 0; i <population.size(); i++)
		{
			population[i].currentPosition();
		}
		sort(population.begin(), population.end(), sortPopulation);

		for (int k = 0; k <= 400; k++)
```

```
{
	if ((k % 100) == 0)
	{
		for (int i = 0; i < 3; i++)
		{
			table.push_back(population[i]);
		}
	}

	population = uniformDistance(population);

	for (int i = 0; i <population.size(); i++)
	{
		population[i].currentPosition();
	}
	sort(population.begin(), population.end(), sortPopulation);

	int min = population[0].getDistance();

	int average = 0;
	for (int i = 0; i <population.size(); i++)
	{
		average += population[i].getDistance();
	}
	average = average / 20;

	file<< max << "\t" <<endl;
		}
	    f              or (int k = 0; k <= 400; k++)
		{
	if ((k % 100) == 0)
	{
		for (int i = 0; i < 3; i++)
		{
			table.push_back(population[i]);
		}
	}

	population = Distance(population);
```