DENIS RAMSKIY II-11

Code of Program

Class of working with gene

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gen_Alg_Lab_2
{
    class Gen
    {
        private List<int> hromosom;
        private double fitness;

        public Gen(Random rnd)
        {
            hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                hromosom.Add(rnd.Next() % 2);

            List<double> temp = new List<double>(parse(hromosom));
            fitness = temp[0] * Math.Sin(Math.Abs(temp[0])) + temp[1] * Math.Sin(Math.Abs(temp[1]));
        }
        public Gen(Gen obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                this.hromosom.Add(obj.get()[i]);

            this.fitness = obj.get_fitness();
        }
        public Gen(List<int> obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                this.hromosom.Add(obj[i]);

            List<double> temp = new List<double>(parse(hromosom));
            fitness = temp[0] * Math.Sin(Math.Abs(temp[0])) + temp[1] * Math.Sin(Math.Abs(temp[1]));
        }

        public void mutation(Random rnd)
        {
            int palace = rnd.Next() % 22;
            if (hromosom[palace] == 1)
                hromosom[palace] = 0;
            else
                hromosom[palace] = 1;
        }
        public List<double> parse(List<int> gn)
        {
            List<double> result = new List<double>();

            string str = "";
            for (int i = 1; i < 11; i++)
                str += gn[i].ToString();
            result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);
            if (gn[0] == 0)
                result[0] *= -1;
```

```
            str = "";
            for (int i = 12; i < 22; i++)
                str += gn[i].ToString();
            result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);
            if (gn[0] == 0)
                result[1] *= -1;

            return result;
        }
        static public Gen operator +(Gen obj1, Gen obj2)
        {
            obj1.get().Clear();
            for (int i = 0; i < 22; i++)
                obj1.get().Add(obj2.get()[i]);
            obj1.set_fitness(obj2.get_fitness());

            return obj1;
        }

        public void set_fitness(double a)
        {
            this.fitness = a;
        }
        public List<int> get()
        {
            return hromosom;
        }
        public double get_fitness()
        {
            return fitness;
        }
    }
}
```

Class of working with generation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Gen_Alg_Lab_2
{
    class Generation
    {
        private List<Gen> generation;
        private double fitness;

        public Generation(Random rnd)
        {
            generation = new List<Gen>();
            for (int I = 0; I < 20; I++)
            {
                Gen temp = new Gen(rnd);
                generation.Add(temp);
                fitness += temp.get_fitness();
            }
            this.sort();
        }
        public void sort()
        {
            for (int i = 0; i < 20; i++)
                for (int j = 0; j < 19; j++)
```

```
                    if (generation[j].get_fitness() < generation[j + 1].get_fitness())
                    {
                        Gen temp = new Gen(generation[j]);
                        generation[j] += generation[j + 1];
                        generation[j + 1] += temp;
                    }
        }

        public List<Gen> get_parents(Random rnd)
        {
            List<Gen> TEMP = new List<Gen>();
            TEMP.Add(generation[rnd.Next() % 10]);
            TEMP.Add(generation[rnd.Next() % 10]);
            return TEMP;
        }
        public Gen get_child(Random rnd, List<Gen> par)
        {
            List<int> TEMP = new List<int>();
            for (int i = 0; i < 22; i++)
                if (rnd.Next() % 2 == 0)
                    TEMP.Add(par[0].get()[i]);
                else
                    TEMP.Add(par[1].get()[i]);

            Gen child = new Gen(TEMP);
            if (rnd.Next() % 1000 < 10)
                child.mutation(rnd);
            return child;
        }
        public void new_generation(Random rnd)
        {
            List<Gen> ngeneration = new List<Gen>();
            double nfitness = 0;
            for (int i = 0; i < 20; i++)
            {
                ngeneration.Add(this.get_child(rnd, this.get_parents(rnd)));
                nfitness += ngeneration[i].get_fitness();
            }

            for (int i = 0; i < 20; i++)
                this.generation[i] += ngeneration[i];
            this.fitness = nfitness;
            this.sort();
        }

        public double get_fitness()
        {
            return fitness;
        }
        public List<Gen> get()
        {
            return generation;
        }
    }
}
```

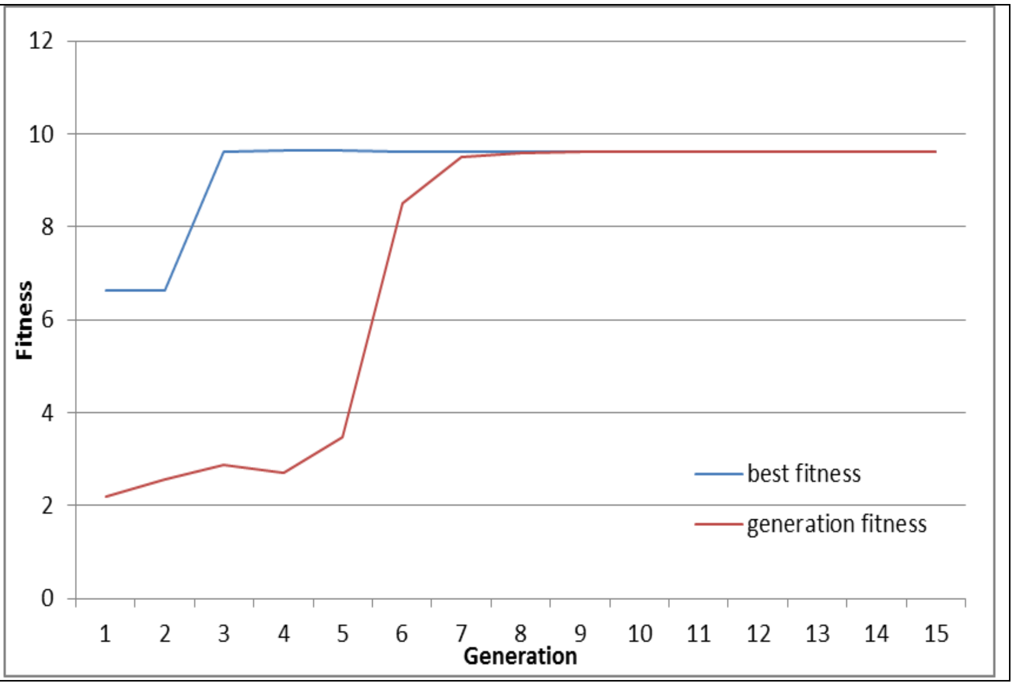
Main program class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
```

```
namespace Gen_Alg_Lab_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random(DateTime.Now.Millisecond);
            Generation work = new Generation(rnd);

            File.WriteAllText("y.txt", "");
            File.WriteAllText("x1.txt", "");
            File.WriteAllText("x2.txt", "");
            File.WriteAllText("fitness.txt", "");
            File.WriteAllText("gen_fitness.txt", "");

            while (true)
            {
                work.new_generation(rnd);
                for (int i = 0; i < 20; i++)
                {
                    File.AppendAllText("y.txt", work.get()[i].get_fitness().ToString() + "\r\n");
                    File.AppendAllText("x1.txt", parse(work.get()[i].get())[0] + "\r\n");
                    File.AppendAllText("x2.txt", parse(work.get()[i].get())[1] + "\r\n");
                }
                File.AppendAllText("fitness.txt", work.get()[0].get_fitness() + "\r\n");
                File.AppendAllText("gen_fitness.txt", work.get_fitness() + "\r\n");

                File.AppendAllText("y.txt", "\r\n\r\n");
                File.AppendAllText("x1.txt", "\r\n\r\n");
                File.AppendAllText("x2.txt", "\r\n\r\n");
            }
        }

        static public List<double> parse(List<int> gn)
        {
            List<double> result = new List<double>();

            string str = "";
            for (int i = 1; i < 11; i++)
                str += gn[i].ToString();
                result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);
                if (gn[0] == 0)
                    result[0] *= -1;

            str = "";
            for (int i = 12; i < 22; i++)
                str += gn[i].ToString();
            result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);
            if (gn[0] == 0)
                result[1] *= -1;

            return result;
        }
    }
}
```

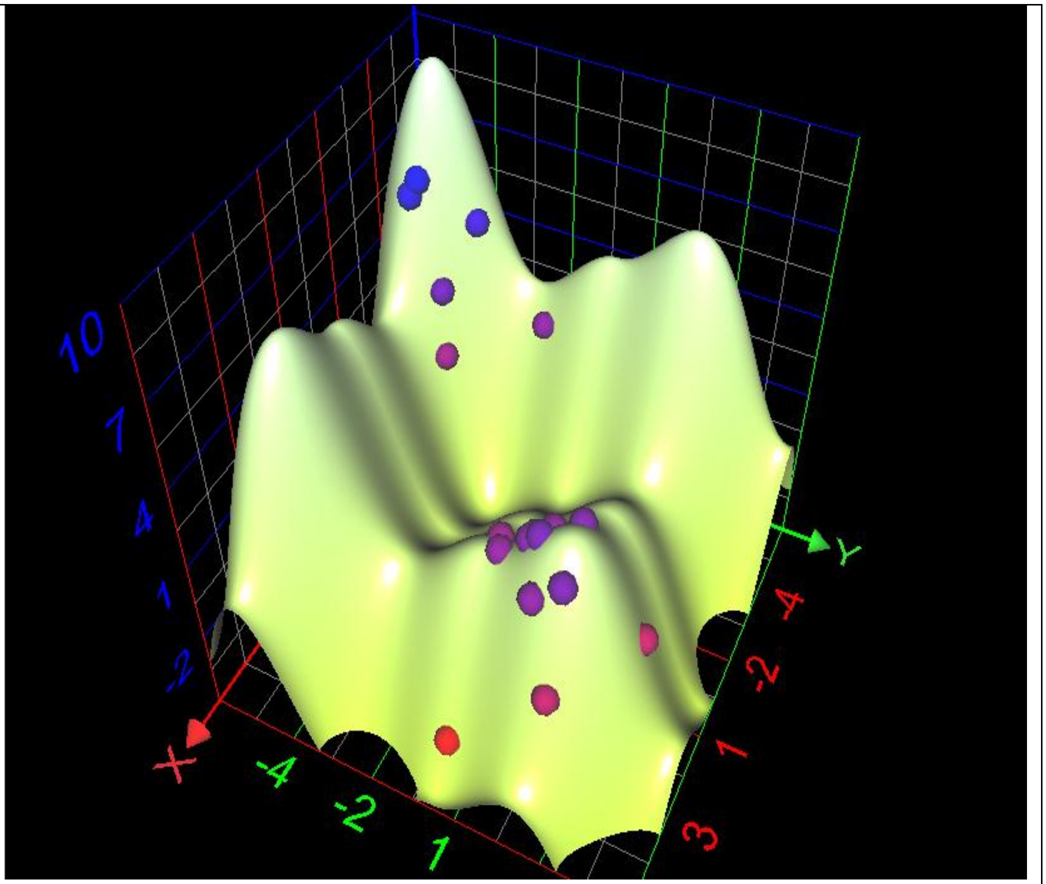
My results:

best fitness	generation fitness
6,635524417	2,1951397
6,635524417	2,572840364
9,60804426	2,882155466
9,627960725	2,713054934
9,627960725	3,469587314
9,604019931	8,493067863
9,604019931	9,513383727
9,609762244	9,582030812
9,611812097	9,60049037
9,611812097	9,603523173
9,611812097	9,606747189
9,611812097	9,609864055
9,611812097	9,611812097
9,611812097	9,611812097
9,611812097	9,611812097



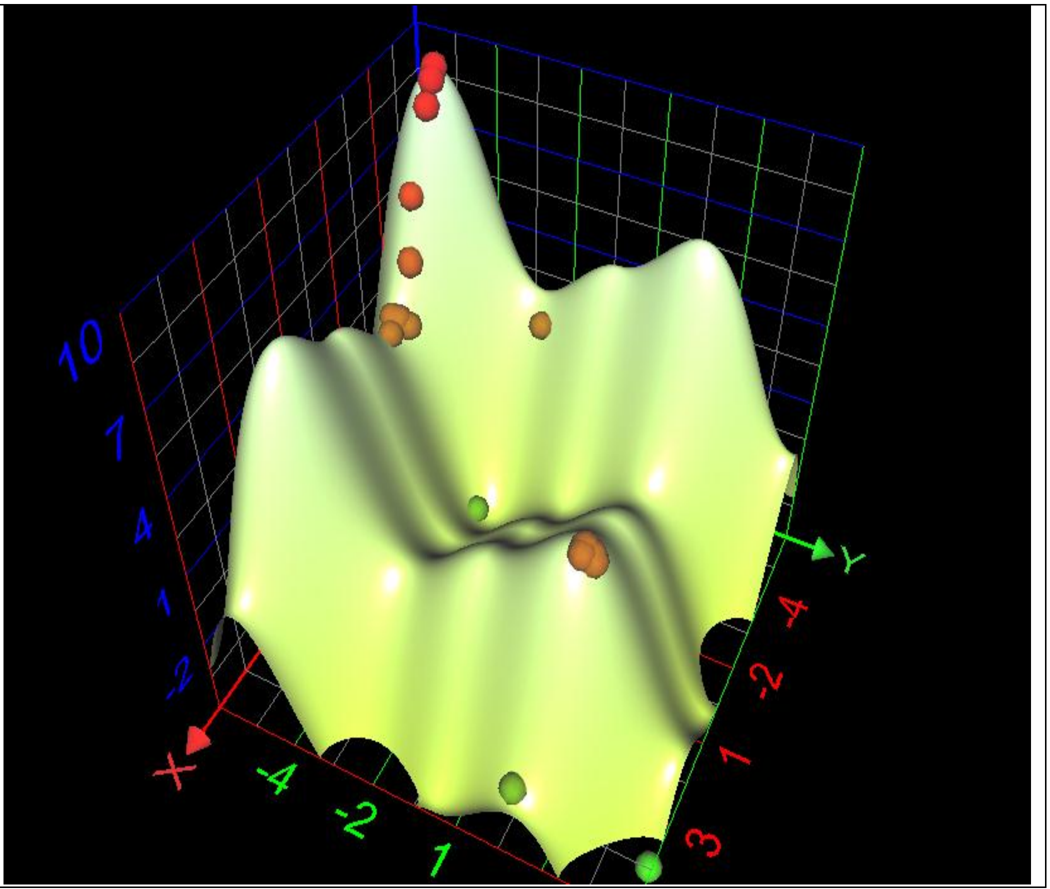
Generation 1

X1	X2	Y
-3,68035	-4,75073	6,635524
-3,53372	-4,89247	6,1638
-4,31574	-3,47996	5,135831
-3,28446	-3,95406	3,33822
1,955034	1,280547	3,039467
2,751711	2,174976	2,835805
1,021505	1,871945	2,658938
1,783969	1,050831	2,655537
1,886608	1,001955	2,637478
-4,24731	-1,55914	2,237158
2,917889	1,549365	2,196322
1,050831	1,197458	2,026921
2,018573	0,337243	1,931155
1,344086	0,024438	1,310289
-2,913	-3,6999	1,299932
0,645161	0,694037	0,831891
0,747801	0,42522	0,683937
3,680352	2,258065	-0,14284
1,383187	3,611926	-0,27795
4,081134	0,004888	-3,29462



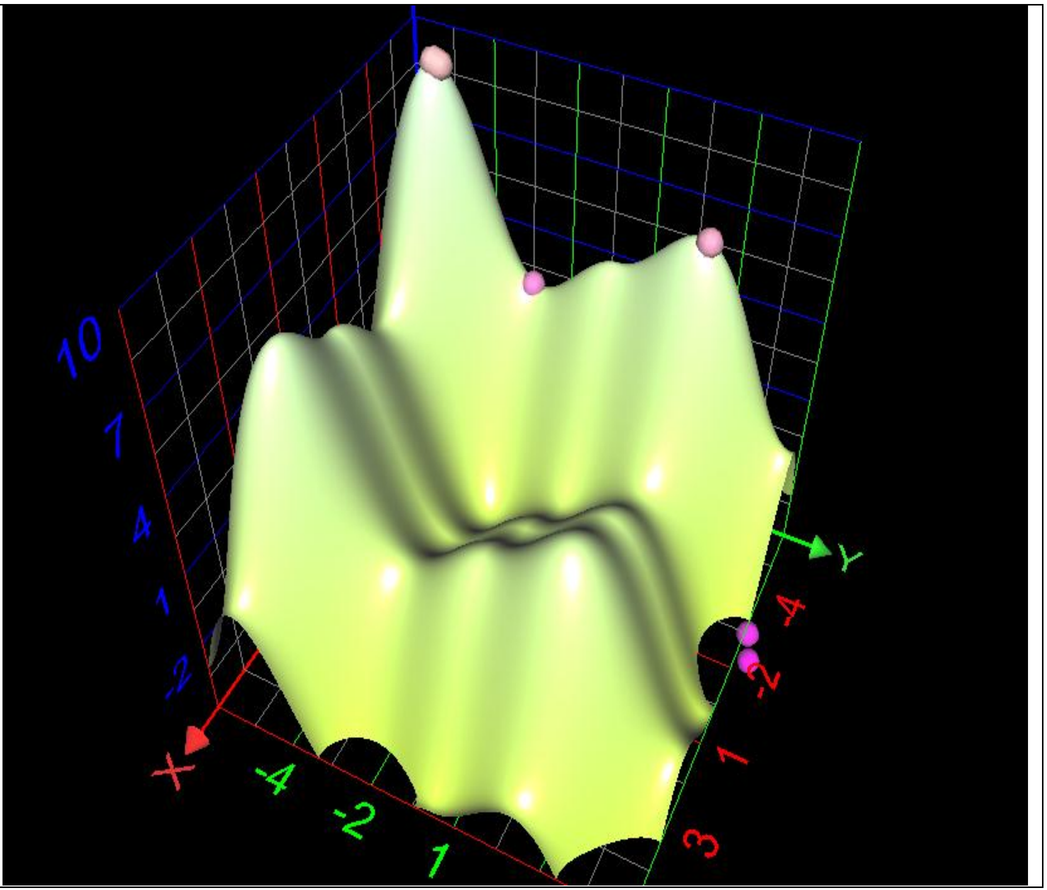
Generation 4

X1	X2	Y
-4,93157	-4,90714	9,627961
-4,65787	-4,83382	9,449174
-4,34506	-4,80938	8,841988
-3,59726	-4,91202	6,397493
-3,09384	-4,75073	4,599563
-3,09384	-4,75073	4,599563
-3,07429	-4,73118	4,523599
1,959922	2,248289	3,565147
2,1261	2,258065	3,552076
1,969697	2,414467	3,420009
2,043011	2,487781	3,332539
-2,34604	-4,90714	3,138715
-2,39003	-4,73118	3,098479
-1,97947	-4,74096	2,922565
-2,46823	-4,50147	2,862473
-4,35484	-1,75953	2,35114
4,545455	1,86217	-2,69859
-2,03812	-2,49756	-3,31919
3,29912	4,765396	-6,39557
4,868035	4,990225	-9,60804



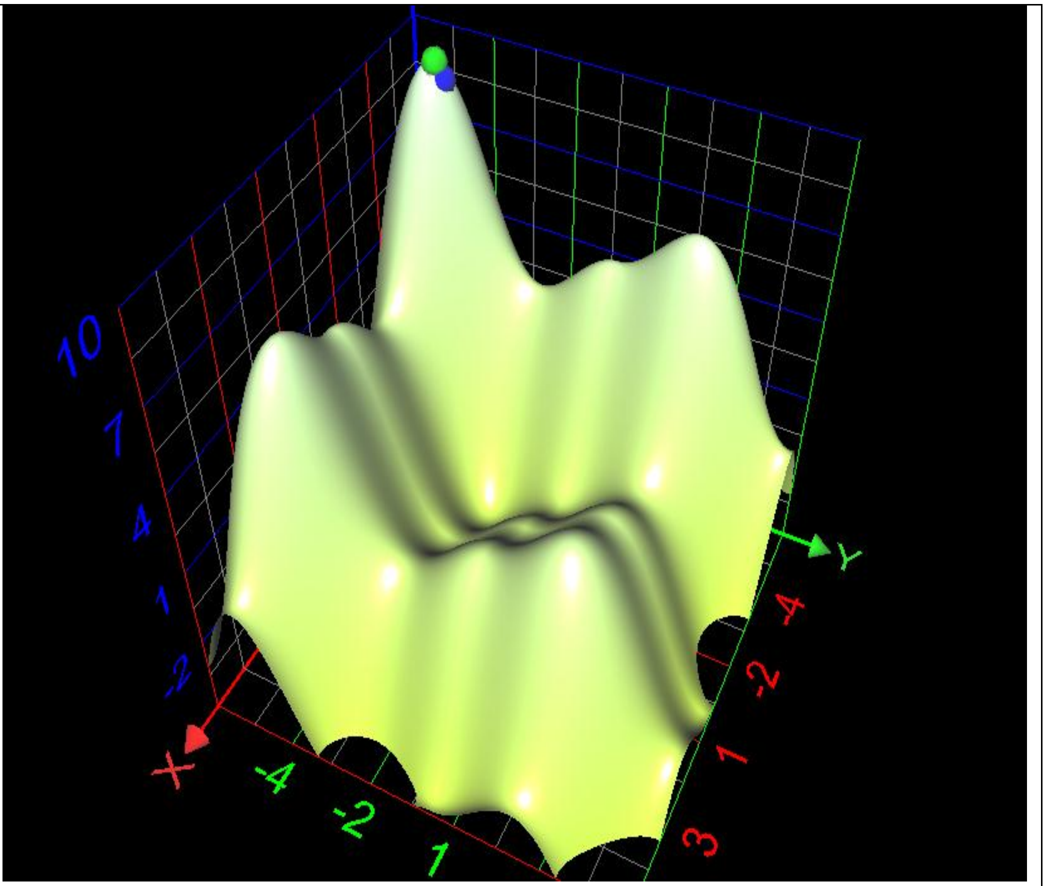
Generation 7

X1	X2	Y
-4,97067	-4,83382	9,60402
-4,97067	-4,82893	9,60197
-4,93157	-4,80938	9,600364
-4,97067	-4,81427	9,595101
-4,97067	-4,81427	9,595101
-4,97067	-4,75073	9,553035
-4,97067	-4,75073	9,553035
-4,97067	-4,75073	9,553035
-4,97067	-4,75073	9,553035
-4,93157	-4,73607	9,548328
-4,93157	-4,73607	9,548328
-4,97067	-4,73607	9,540536
-4,97067	-4,73118	9,536141
-4,97067	-4,73118	9,536141
-4,97067	-4,73118	9,536141
-4,65787	-4,75073	9,398189
-4,61877	-4,82893	9,394718
-4,61877	-4,75073	9,345783
-4,61877	-4,75073	9,345783
-4,61877	-4,73118	9,328889



Generation 11

X1	X2	Y
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,82893	9,609762
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,97067	-4,83382	9,60402
-4,93157	-4,81427	9,602893
-4,93157	-4,81427	9,602893
-4,97067	-4,82893	9,60197
-4,97067	-4,82893	9,60197
-4,97067	-4,82893	9,60197
-4,97067	-4,82893	9,60197
-4,97067	-4,82893	9,60197
-4,93157	-4,80938	9,600364
-4,93157	-4,80938	9,600364
-4,97067	-4,80938	9,592572



Generation 15

X1	X2	Y
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812
-4,93157	-4,83382	9,611812