Zykov Kirill. Laboratory work №2.
So, I have finished neural network part of the program but since I cant use code from previous laboratory work because different data is used, I wasn't able to finish my program.

Code:

```
class Program
    {
        private static Random rnd = new Random(DateTime.Now.Millisecond);
        static void Main(string[] args)
        {
            List<List<int>> input = new List<List<int>>();
            input = getBNumbers();
            for (int i = 0; i < 64; i++)
            {
                for (int j = 0; j < 6; j++)
                {
                    Console.Write(input[i][j].ToString());
                }
                Console.WriteLine();
            }
            //List<List<double>> w1 = new List<List<double>>();
            List<double> w = new List<double>();
            List<int> output = new List<int>();
            List<int> etalon = new List<int>();
            etalon = getEtalon(input);
            for (int i = 0; i < 42; i++)
            {
                w.Add(Convert.ToDouble(rnd.Next(0, 2) * 2 - 1) * rnd.NextDouble());
            }
            List<List<double>> w1 = new List<List<double>>();
            for (int i = 0; i < 6; i++)
            {
                List<double> temp = new List<double>();
                for (int j = 0; j < 6; j++)
                {
                    temp.Add(Convert.ToDouble(rnd.Next(0, 2) * 2 - 1) *
rnd.NextDouble());
                }
                w1.Add(temp);
            }
            List<double> w2 = new List<double>();
            for (int i = 0; i < 6; i++)
            {
                w2.Add(Convert.ToDouble(rnd.Next(0, 2) * 2 - 1) * rnd.NextDouble());
            }


        }

        public static List<List<int>> getBNumbers()
        {
            List<List<int>> bNumbers = new List<List<int>>();
            for (int i = 0; i < 64; i++)
            {
                if (Convert.ToString(i, 2).Length != 6)
                {
                    string temp = "";
                    for (int j = 0; j < 6 - Convert.ToString(i, 2).Length; j++)
                    {
                        temp += "0";
                    }
```

```
                temp += Convert.ToString(i, 2);
                List<int> tempInt = new List<int>();
                for (int k = 0; k < 6; k++)
                {
                    if (temp[k] == '0') { tempInt.Add(-1); }
                    else { tempInt.Add(1); }
                }
                bNumbers.Add(tempInt);

            }
            else
            {
                string temp = Convert.ToString(i, 2);
                List<int> tempInt = new List<int>();
                for (int k = 0; k < 6; k++)
                {
                    if (temp[k] == '0') { tempInt.Add(-1); }
                    else { tempInt.Add(1); }
                }
                bNumbers.Add(tempInt);
            }
        }
        return bNumbers;
    }
    public static List<int> getEtalon(List<List<int>> input)
    {
        List<int> etalon = new List<int>();
        for (int i = 0; i < input.Count; i++)
        {
            int temp = 0;
            for (int j = 0; j < input[i].Count; j++)
            {
                if (input[i][j] != -1)
                {
                    temp += input[i][j];
                }
            }
            if (temp % 2 == 0)
            {
                etalon.Add(1);
            }
            else
            {
                etalon.Add(-1);
            }
            Console.WriteLine(etalon[i]);
        }
        return etalon;
    }
    public static int neuroOutput(List<int> x,  List<double> w)
    {
        double y_d = 0;
        int y = 0;
        int c = 0;
        List<double> o1 = new List<double>();
        for (int i = 0; i < 6; i++)
        {
            double temp = 0.0;
            for (int j = 0; j < 6; j++)
            {
                temp += x[i] * w[c];
                c++;
            }
```

```
                o1.Add(temp);
            }
            for (int i = 0; i < 6; i++)
            {
                y_d += o1[i] * w[c];
                c++;
            }
            if (y_d < 0) { y = -1; }
            else { y = 1; }
            return y;
        }

        public static List<int> fullNeuroOutput(List<List<int>> input, List<double>
w)
        {
            List<int> yList = new List<int>();
            for (int i = 0; i < 64; i++)
            {
                yList.Add(neuroOutput(input[i], w));
            }
            return yList;
        }
        public static int getFittnes(List<int> input, List<int> etalon)
        {
            int fittnes = 0;
            for (int i = 0; i < 64; i++)
            {
                if (input[i] == etalon[i]) { fittnes++; }
            }
            return fittnes;
        }

        public static List<double> onePointCross(double a1, double a2)
        {
            List<double> output = new List<double>();
            BitArray BAA1 = new BitArray(BitConverter.GetBytes(a1));
            BitArray BAA2 = new BitArray(BitConverter.GetBytes(a2));
            for (int i = rnd.Next(0,64); i > 0; i--)
            {
                var temp1 = BAA1.Get(i);
                var temp2 = BAA2.Get(i);

                BAA1.Set(i, temp2);
                BAA2.Set(i, temp1);


            }
            byte[] tempbytes = new byte[BAA1.Length];

            BAA1.CopyTo(tempbytes, 0);
            double baa1 = BitConverter.ToDouble(tempbytes, 0);

            BAA2.CopyTo(tempbytes, 0);
            double baa2 = BitConverter.ToDouble(tempbytes, 0);
            output.Add(baa1);
            output.Add(baa2);
            return output;
        }
        public List<List<double>> Selection(List<List<int>> pop, List<int> etalon)
        {
            List<List<double>> newPop = new List<List<double>>();
            List<int> pop_fit = new List<int>();
            for (int i = 0; i < pop.Count; i++)
```

```
            {
                pop_fit.Add(getFittnes(pop[i],etalon));
            }

            SortedList<int, List<int>> sorted = new SortedList<int, List<int>>();
            for (int i = 0; i < pop.Count; i++)
            {
                sorted.Add(pop_fit[i], pop[i]);
            }
            pop = pop.OrderBy(x => pop_fit.IndexOf(pop_fit[x])).ToList();
            List<Chromosome> childrens = new List<Chromosome>();
            for (int i = 0; i < population.Count; i++)
            {
                childrens.Add(
                    population[rnd.Next(halfPop,
popSize)].OnePointCross(population[rnd.Next(halfPop, popSize)], Rnd(rnd, geneNum))
                    );
            }
            childrens[Rnd(rnd, geneNum)].Mutation();
            //childrens = SortPop(childrens);
            newPop = new Population(childrens);
            return newPop;
        }


    }
```