

# Task 3

## Neural Network for Even-n-Parity

### Code:

```
using System;

using System.Collections.Generic;

using System.Linq;

namespace ConsoleApplication1
{
    public class Network
    {
        private double[,] weight = new double[generation, genes];
        private const int genes = 30;
        private const int generation = 100;
        private int[, ] y = new int[100, 32];
        int[, ] p = new int[100, 32];
        private int[] fitness = new int[100];
        private int[] e = new int[];
        private int[, ] inputs = new int [, ] ;

        public Network()
        {
            Random rand = new Random();
            for (int i = 0; i < generation; i++)
            {
                for (int j = 0; j < genes; j++)
                {
                    weight[i, j] = rand.Next(-1, 2) * rand.NextDouble();
                }
            }
        }
    }
}
```

```

    }
}

int count = 0;
for (int i = 0; i < 100; i++)
{
    count = 0;
    for (int j = 0; j < 32; j++)
    {
        y[i, j] = Calculate(j);
        if (y[i, j] == e[j])
            count++;
    }
    fitness[i] = count;
}

}

public int Calculate(int h)
{
    double result = 0;
    int[] res = new int[5];

    for (int i = 0; i < 5; i++)
    {
        result = 0;
        for (int j = 0; j < 30; j++)
        {
            result += inputs[h, i] * weight[i, j];
        }
    }
}

```

```

        if (result >= 0)
            res[i] = 1;
        else
            res[i] = -1;
    }
    for (int i = 0; i < 5; i++)
    {
        result = 0;
        for (int j = 0; j < 30; j++)
        {
            result += res[i] * weight[i, j];
        }
    }
    int r = 0;
    if (result >= 0)
        r = 1;
    else
        r = -1;

    return r;
}

```

```

public void Cross()
{
    int[] keys = new int[32];
    int[,] newPopulation = new int[100 / 2, 32];
    for (int i = 0; i < 32; i++)
        keys[i] = i;
    Array.Sort(fitness, keys);
    for (int i = 0; i < 100 / 2; i++)

```

```

{
    for (int j = 0; j < 32; j++)
    {
        newPopulation[i, j] = y[keys[i], j];
    }
}

Random rand = new Random();
for (int i = 0; i < 100; i++)
{
    int a = rand.Next(0, 100 / 2);
    int b = rand.Next(0, 100 / 2);
    for (int j = 0; j < 32; j++)
    {
        p[0, j] = newPopulation[a, j];
        p[1, j] = newPopulation[b, j];
    }

    for (int j = 0; j < 32; j++)
    {
        int c = rand.Next(0, 2);
        if (c == 0)
        {
            y[i, j] = p[0, j];
        }
        else
        {
            y[i, j] = p[1, j];
        }
    }
}
}

```

```
}
```

```
public int Get_fitness()
```

```
{
```

```
    return fitness.Max();
```

```
}
```

```
public void Mutation()
```

```
{
```

```
    Random rand = new Random();
```

```
    for (int i = 0; i < 100; i++)
```

```
    {
```

```
        for (int j = 0; j < 32; j++)
```

```
        {
```

```
            int value = rand.Next(0, 32);
```

```
            if (value == 0)
```

```
            {
```

```
                weight[i, j] = rand.Next(-1, 2) * rand.NextDouble();
```

```
            }
```

```
        }
```

```
    }
```

```
    int count = 0;
```

```
    for (int i = 0; i < 100; i++)
```

```
    {
```

```
        count = 0;
```

```
        for (int j = 0; j < 32; j++)
```

```
        {
```

```
            y[i, j] = Calculate(j);
```

```

        if (y[i, j] == inputs[i, j])
            count++;
    }
    fitness[i] = count;
}
}
}
}
}

```

```

//*****

```

```

using System;

```

```

namespace ConsoleApplication1

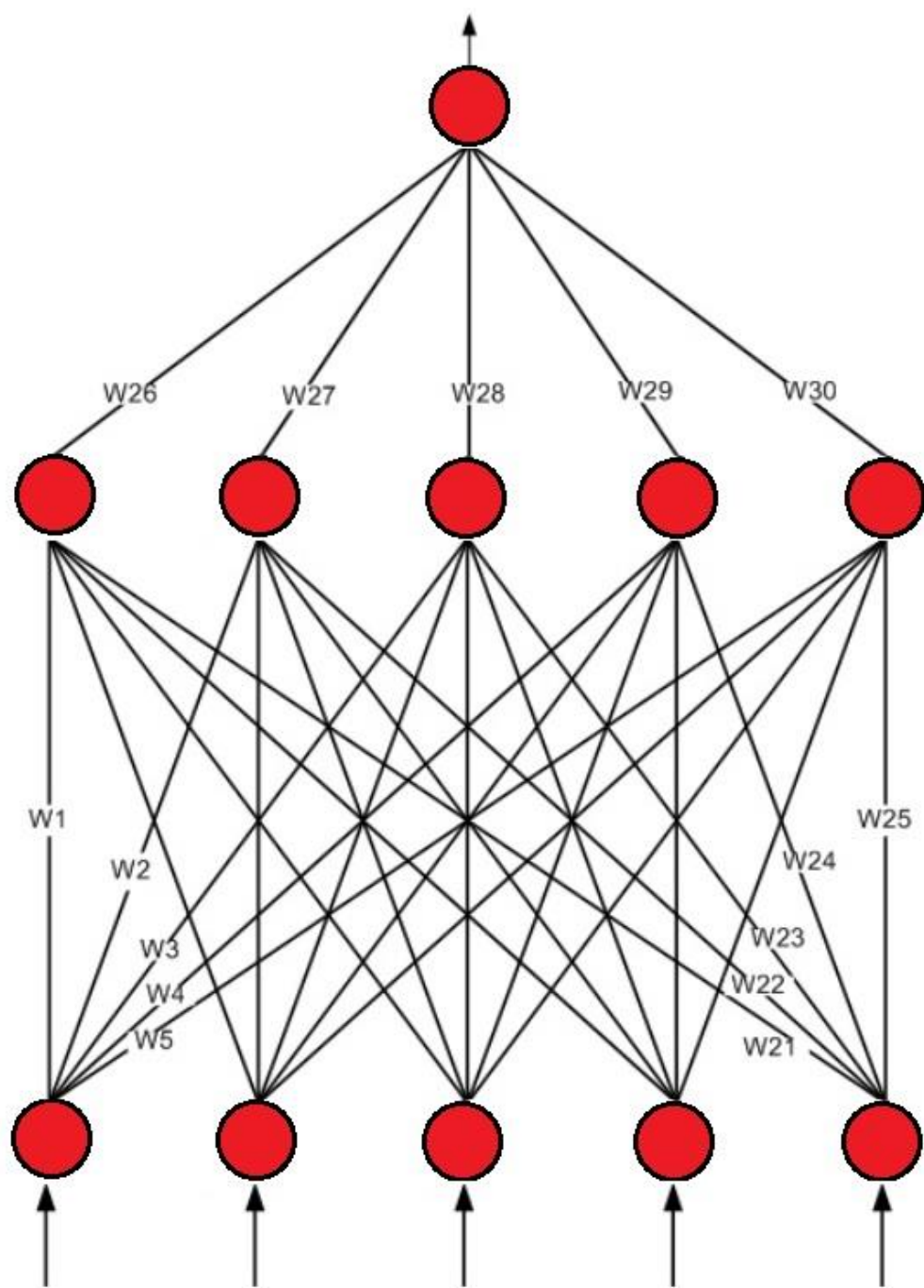
```

```

{
    class Program
    {
        public class Conditional
        {
            public static bool Condit(int max)
            {
                if (max == 32)
                {
                    return false;
                }
                else
                    return true;
            }
        }
    }
}

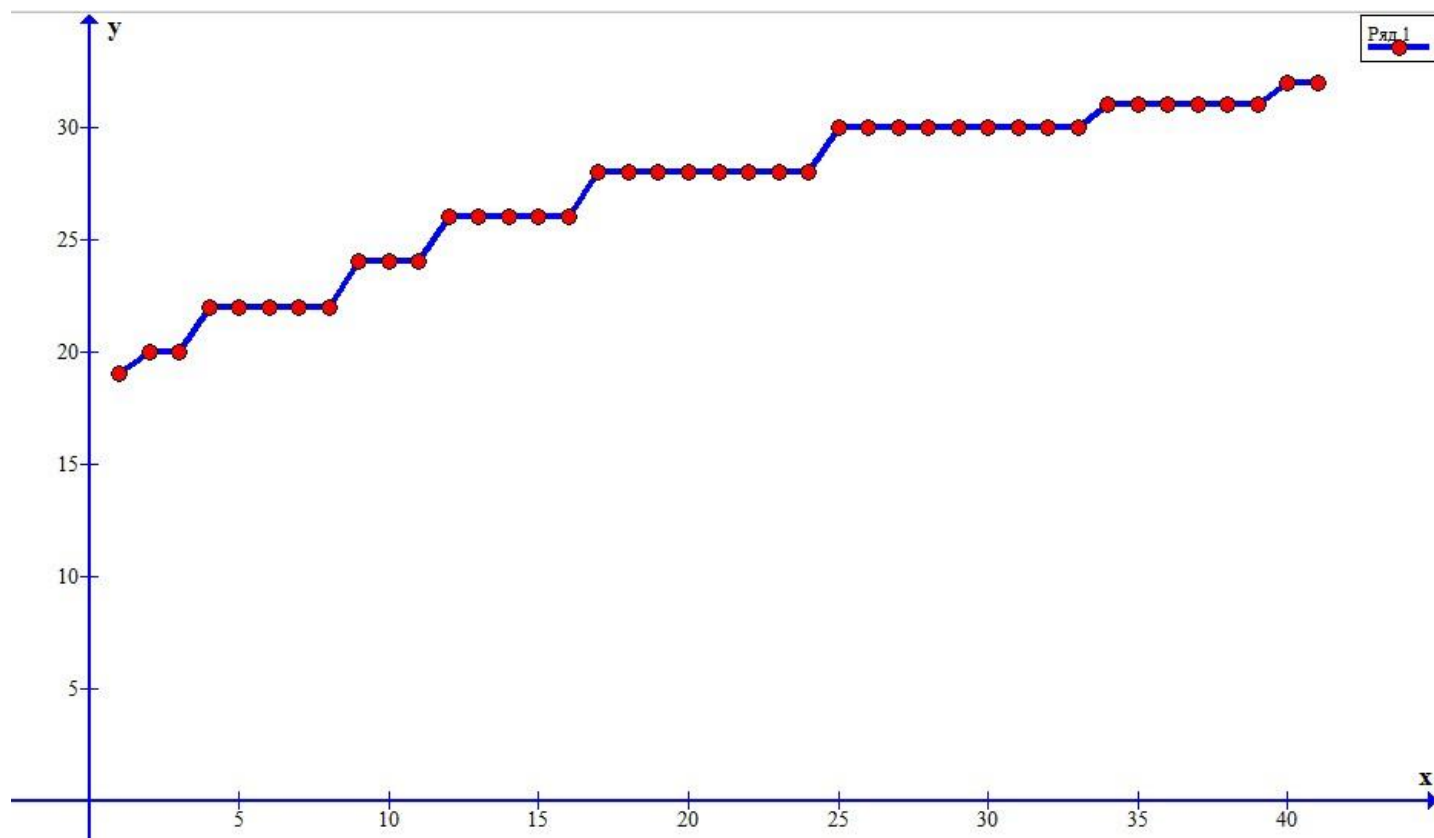
```

```
static void Main(string[] args)
{
    Network nw = new Network();
    int max = nw.Get_fitness();
    while (Conditional.Condit(max))
    {
        nw.Cross();
        nw.Mutation();
        max = nw.Get_fitness();
        Console.WriteLine(max);
    }
}
}
```





| x1 | x2 | x3 | x4 | x5 | y  |
|----|----|----|----|----|----|
| 1  | 1  | 1  | 1  | 1  | -1 |
| 1  | 1  | 1  | 1  | -1 | 1  |
| 1  | 1  | 1  | -1 | 1  | 1  |
| 1  | 1  | 1  | -1 | -1 | -1 |
| 1  | 1  | -1 | 1  | 1  | 1  |
| 1  | 1  | -1 | 1  | -1 | -1 |
| 1  | 1  | -1 | -1 | 1  | -1 |
| 1  | 1  | -1 | -1 | -1 | 1  |
| 1  | -1 | 1  | 1  | 1  | 1  |
| 1  | -1 | 1  | 1  | -1 | -1 |
| 1  | -1 | 1  | -1 | 1  | -1 |
| 1  | -1 | 1  | -1 | -1 | 1  |
| 1  | -1 | -1 | 1  | 1  | -1 |
| 1  | -1 | -1 | 1  | -1 | 1  |
| 1  | -1 | -1 | -1 | 1  | 1  |
| 1  | -1 | -1 | -1 | -1 | -1 |
| -1 | 1  | 1  | 1  | 1  | 1  |
| -1 | 1  | 1  | 1  | -1 | -1 |
| -1 | 1  | 1  | -1 | 1  | -1 |
| -1 | 1  | 1  | -1 | -1 | 1  |
| -1 | 1  | -1 | 1  | 1  | -1 |
| -1 | 1  | -1 | 1  | -1 | 1  |
| -1 | 1  | -1 | -1 | 1  | 1  |
| -1 | 1  | -1 | -1 | -1 | -1 |
| -1 | -1 | 1  | 1  | 1  | -1 |
| -1 | -1 | 1  | 1  | -1 | 1  |
| -1 | -1 | 1  | -1 | 1  | 1  |
| -1 | -1 | 1  | -1 | -1 | -1 |
| -1 | -1 | -1 | 1  | 1  | 1  |
| -1 | -1 | -1 | 1  | -1 | -1 |
| -1 | -1 | -1 | -1 | 1  | 1  |
| -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1  | -1 |
| -1 | -1 | -1 | -1 | -1 | 1  |



| Weight | Value    |
|--------|----------|
| w1     | -0,60859 |
| w2     | 0,33773  |
| w3     | 0,10569  |
| w4     | 0,19265  |
| w5     | -0,32895 |
| w6     | -0,30501 |
| w7     | -0,12658 |
| w8     | 0,24123  |
| w9     | 0,45993  |
| w10    | 0,32654  |
| w11    | 0,45738  |
| w12    | -0,43941 |
| w13    | -0,09348 |
| w14    | -0,74685 |
| w15    | 0,19476  |
| w16    | 0,42447  |
| w17    | -0,89497 |
| w18    | -0,71541 |
| w19    | -0,94342 |
| w20    | -0,21546 |
| w21    | 0,86668  |
| w22    | -0,90866 |
| w23    | 0,74833  |
| w24    | 0,52245  |
| w25    | -0,21497 |
| w26    | -0,72596 |
| w27    | -0,04055 |
| w28    | 0,17638  |
| w29    | -0,25267 |
| w30    | -0,53186 |

| Generation | 1  | 10 | 20 | 30 | 41 |
|------------|----|----|----|----|----|
| Fitness    | 19 | 24 | 28 | 30 | 32 |
|            | 1  | -1 | -1 | -1 | -1 |
|            | 1  | 1  | 1  | 1  | 1  |
|            | -1 | 1  | -1 | 1  | 1  |
|            | 1  | 1  | -1 | -1 | -1 |
|            | 1  | -1 | 1  | 1  | 1  |
|            | 1  | 1  | 1  | 1  | -1 |
|            | -1 | 1  | -1 | -1 | -1 |
|            | 1  | -1 | 1  | 1  | 1  |
|            | -1 | 1  | -1 | 1  | 1  |
|            | 1  | -1 | 1  | -1 | -1 |
|            | -1 | 1  | -1 | 1  | -1 |
|            | 1  | 1  | 1  | 1  | 1  |
|            | 1  | -1 | 1  | -1 | -1 |
|            | 1  | 1  | 1  | -1 | 1  |
|            | 1  | -1 | -1 | 1  | 1  |
|            | -1 | -1 | -1 | 1  | -1 |
|            | 1  | 1  | -1 | 1  | 1  |
|            | 1  | 1  | -1 | -1 | -1 |
|            | 1  | 1  | 1  | -1 | -1 |
|            | -1 | 1  | 1  | 1  | 1  |
|            | -1 | -1 | 1  | -1 | -1 |
|            | -1 | 1  | 1  | 1  | 1  |
|            | 1  | 1  | -1 | 1  | 1  |
|            | 1  | 1  | -1 | -1 | -1 |
|            | -1 | -1 | 1  | -1 | -1 |
|            | 1  | 1  | 1  | 1  | 1  |
|            | -1 | -1 | 1  | 1  | 1  |
|            | 1  | -1 | -1 | -1 | -1 |
|            | 1  | 1  | 1  | 1  | 1  |
|            | 1  | 1  | 1  | -1 | -1 |
|            | 1  | -1 | 1  | 1  | -1 |
|            | -1 | -1 | 1  | 1  | 1  |