```
class Program
    {
        static void Main()
        {
            int k = int.Parse(Console.ReadLine());
            int n = int.Parse(Console.ReadLine());
            int len = 0;
            int lastBit = -1;
            int dancingBitsCount = 0;
            for (int i = 0; i < n; i++)
            {
                int num = int.Parse(Console.ReadLine());

                // Skip the leading zeroes in num
                int firstNonZeroBit = 0;
                for (int bitNum = 31; bitNum >= 0; bitNum--)
                {
                    int currentBit = (num >> bitNum) & 1;
                    if (currentBit == 1)
                    {
                        firstNonZeroBit = bitNum;
                        break;
                    }
                }

                // Process the digits of num (without the leading zeroes)
                for (int bitNum = firstNonZeroBit; bitNum >= 0; bitNum--)
                {
                    int currentBit = (num >> bitNum) & 1;
                    if (currentBit == lastBit)
                    {
                        // The current bits continues the last sequence
                        len++;
                    }
                    else
                    {
                        // The sequence is changed --> start a new sequence
                        if (len == k)
                        {
                            dancingBitsCount++;
                        }
                        len = 1;
                    }
                    lastBit = currentBit;
                }
            }

            // Check the last sequence
            if (len == k)
            {
                dancingBitsCount++;
            }

            Console.WriteLine(dancingBitsCount);
        }
    }

using System;

namespace Problem_5_Lines
{
    class Program
    {
```

```
static void Main()
{
    // Read the input numbers
    int num0 = Int32.Parse(Console.ReadLine());
    int num1 = Int32.Parse(Console.ReadLine());
    int num2 = Int32.Parse(Console.ReadLine());
    int num3 = Int32.Parse(Console.ReadLine());
    int num4 = Int32.Parse(Console.ReadLine());
    int num5 = Int32.Parse(Console.ReadLine());
    int num6 = Int32.Parse(Console.ReadLine());
    int num7 = Int32.Parse(Console.ReadLine());

    int bestLen = 0;
    int bestCount = 0;

    // Check all horizontal lines
    for (int row = 0; row <= 7; row++)
    {
        int rowBits = 0;
        switch (row)
        {
            case 0: rowBits = num0; break;
            case 1: rowBits = num1; break;
            case 2: rowBits = num2; break;
            case 3: rowBits = num3; break;
            case 4: rowBits = num4; break;
            case 5: rowBits = num5; break;
            case 6: rowBits = num6; break;
            case 7: rowBits = num7; break;
        }

        int len = 0;
        for (int col = 0; col <= 7; col++)
        {
            int cell = (rowBits >> col) & 1;
            if (cell == 1)
            {
                len++;
                if (len > bestLen)
                {
                    bestLen = len;
                    bestCount = 0;
                }
                if (len == bestLen)
                {
                    bestCount++;
                }
            }
            else
            {
                len = 0;
            }
        }
    }

    // Check all vertical lines
    for (int col = 0; col <= 7; col++)
    {
        int len = 0;
        for (int row = 0; row <= 7; row++)
        {
            int rowBits = 0;
            switch (row)
```

```
                {
                    case 0: rowBits = num0; break;
                    case 1: rowBits = num1; break;
                    case 2: rowBits = num2; break;
                    case 3: rowBits = num3; break;
                    case 4: rowBits = num4; break;
                    case 5: rowBits = num5; break;
                    case 6: rowBits = num6; break;
                    case 7: rowBits = num7; break;
                }

                int cell = (rowBits >> col) & 1;
                if (cell == 1)
                {
                    len++;
                    if (len > bestLen)
                    {
                        bestLen = len;
                        bestCount = 0;
                    }
                    if (len == bestLen)
                    {
                        bestCount++;
                    }
                }
                else
                {
                    len = 0;
                }
            }
        }

        // Check for the special case when the largest cell has size 1x1
        if (bestLen == 1)
        {
            // Cells with size 1x1 were counted twice --> recalculate them
            bestCount = bestCount / 2;
        }

        Console.WriteLine(bestLen);
        Console.WriteLine(bestCount);
    }
}
}
```