PROGRAMMING
B3 DOCUMENTATION

Made by: Yildirim Ammar
Neptun code: FA27QM
E-mail: yildirimammar@gmail.com

Course code: GROUP 2
Teacher's name: MENYHÁRT LÁSZLÓ

2022. December 11.

Content

User documentation

Task

Some people are playing a gambling game. They all say a number between 1 and M. The winner is the person who first says the number that was said by the most of the players. If there is more than one such number, then all the players will win who were the first for such a “most frequent number”.

Write a program that gives the winner player, the number the player won with, and the count of people who said this number.

Runtime environment

A Dell Laptop that is capable of running exe files, 64-bit operating system (eg. Windows 11). No mouse needed.

Usage

Starting the program

The program can be found in the archived file by the name `FA27QM\bin\Release\FA27QM.exe`. You can start the program by clicking the `FA27QM.exe` file.

Program input

The program reads the input data from the keyboard in the following order:

#	Data	Explanation
1.	N	The count of players ($1 \leq N \leq 100$)
2.	M	Maximum limit for the input number ($1 \leq M \leq 1000000$)
3.	$data_1$	The first number given by the first player ($1 \leq data_1 \leq M$)
4.	$data_2$	The second number given by the second player($1 \leq data_2 \leq M$)
..	..	..
N+.	$data_N$	The N^{th} number ($1 \leq data_N \leq M$)

Program output

The standard output should consist of as many lines as many winners there are (in increasing order of the winner number). Each line should contain 3 numbers: the index of the winner, the winning number, and the count of people who chose this number.

Sample input and output

```
=== Input ===
5 1000
1
2
3
4
5
=== Output ===
1 1 1
2 2 1
3 3 1
4 4 1
5 5 1
```

Possible errors

The input should be given according to the sample, which means the following:

N should be given as a whole number, which ranges from 1 to 100. If the input is different, the program will display an error message, and it will ask you to input N and M again.

M should be given as a whole number, which ranges from 1 to 1000000. If the input is different, the program will display an error message, and it will ask you to input N and M again.

Data should be given as a whole number, which ranges from 1 to M. If the input is different, the program will display an error message, and it will ask you to input the last data again.

Sample of running in the case of invalid data:

```
=== Input ===
N=? M=?
hi b
Error! Wrong input for N. Please Input N & M again
Error! Wrong input for M. Please Input N & M again
N=? M=?
-1 10
Error! Wrong input for N. Please Input N & M again
N=? M=?
10 -2
Error! Wrong input for M. Please Input N & M again
N=? M=?
5 100
data 1: 1
data 2: 2
data 3: a
Error! Wrong input for Data. Please input data 3 again
data 3: -1
Error! Wrong input for Data. Please input data 3 again
data 3: 3
data 4: 4
data 5: 5
=== Output ===
1 1 1
2 2 1
3 3 1
4 4 1
5 5 1
```

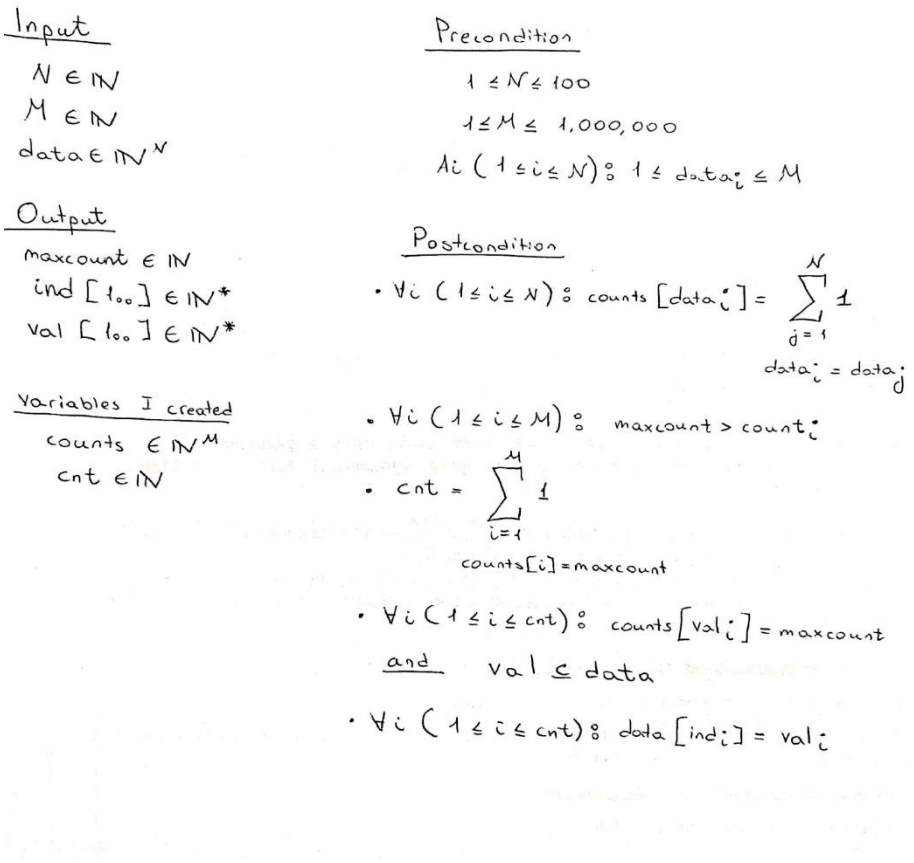
Developer documentation

Task

Some people are playing a gambling game. They all say a number between 1 and M. The winner is the person who first says the number that was said by the most of the players. If there is more than one such number, then all the players will win who were the first for such a “most frequent number”.

Write a program that gives the winner player, the number the player won with, and the count of people who said this number.

Specification



Developer environment

Dell Laptop, an operating system capable of running .exe files (eg. Windows 11). .NET (v6.0.403) for compiling and running and Visual Studio Code (v1.73.1) developer tool for the development environment.

Source code

All the sources can be found in the `FA27QM` folder (after extraction). The folder structure used for development:

File	Explanation
`FA27QM\bin\Release\FA27QM.exe`	Executable code

FA27QM\program.cs	C# source code
FA27QM\test1.txt	input test file$_1$
FA27QM\test2.txt	input test file$_2$
FA27QM\test3.txt	input test file$_3$
FA27QM\test4.txt	input test file$_4$
FA27QM\test5.txt	input test file$_5$
FA27QM\doc\FA27QM.docx	documentation (this file)

Solution

Program parameters

Variables

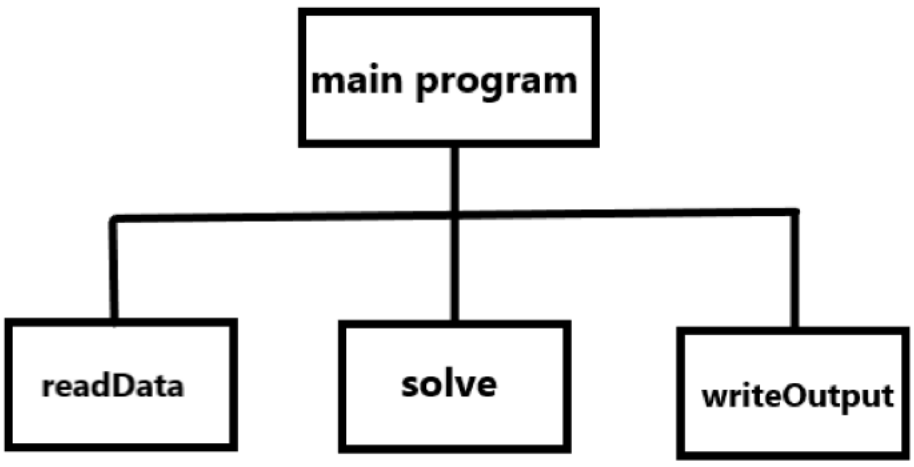
```
N               : Integer
M               : Integer
maxcount        : Integer
cnt             : Integer

Data            : Array(1..N:Integer)
counts          : Array(1..M:Integer)
val             : Array(1..cnt:Integer)
ind             : Array(1..cnt:Integer)
```

The structure of the program

The modules used by the program, and their locations:

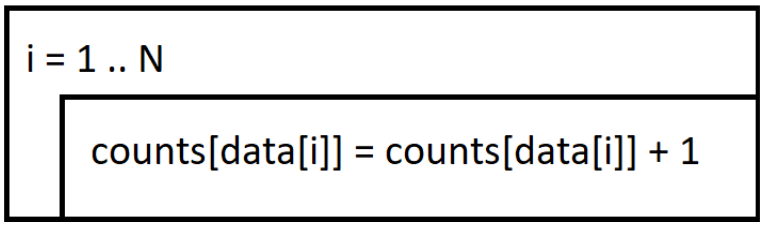
`Program.cs` – the program, in the source folder
`System` –means the program using the C# System library

Structure of functions

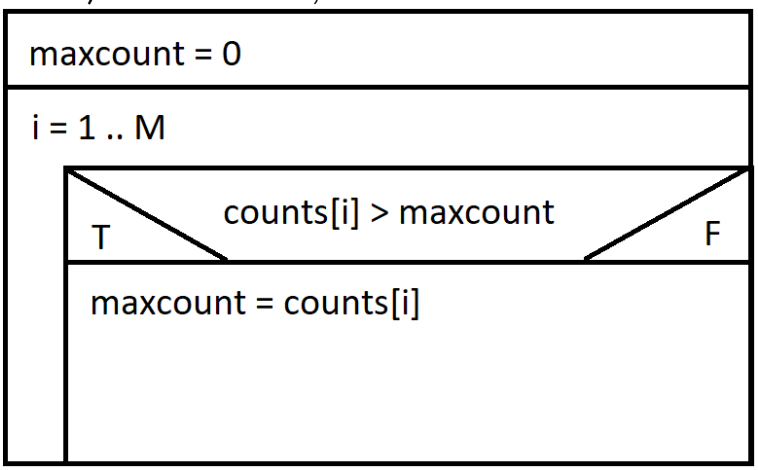


The algorithm of the program

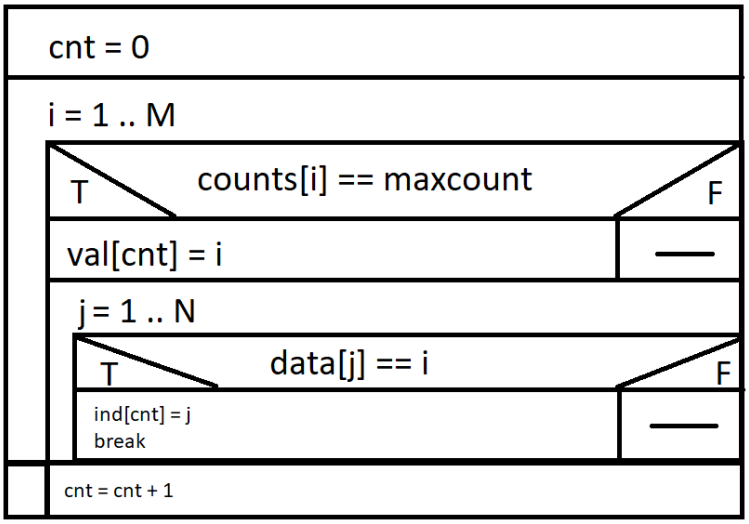
Part 1) Create an array which contains the count of the input numbers



Part 2) Find maxcount, which is the the maximum occurrence a number can have



Part 3) Store the values and indexes of the numbers that have occurred “maxcount” times



The code

The content of the `program.cs` file:

```
using System;

namespace homework3
{
    internal class Program
    {
        static void readData(out int N, out int M, out int[] data)
        {
            bool errorN, errorM, errorData;
```

```
            Console.Error.WriteLine("=== Input ===");
            do
            {
                Console.WriteLine("N=? M=?");
                string[] tmp;
                tmp = Console.ReadLine().Split(" ");
                errorN = (!Int32.TryParse(tmp[0], out N) || (N < 1) || (N > 1000000));
                errorM = (!Int32.TryParse(tmp[1], out M) || (M < 1) || (M > 1000000));
                if (errorN)
                {
                    Console.Error.WriteLine("Error! Wrong input for N. Please Input N
& M again");
                }
                if (errorM)
                {
                    Console.Error.WriteLine("Error! Wrong input for M. Please Input N
& M again");
                }
            } while (errorN || errorM);

            data = new int[N];
            for (int i = 0; i < N; i++)
            {
                do
                {
                    Console.Write("data " + (i+1) + ": ");
                    string tmp = Console.ReadLine();
                    errorData = (!Int32.TryParse(tmp, out data[i]) || data[i] > M ||
data[i] < 1);
                    if (errorData)
                    {
                        Console.Error.WriteLine("Error! Wrong input for Data. Please
input data " + (i+1) + " again");
                    }
                } while (errorData);
            }
        }

        static void solve(int N, int M,int[] data, out int cnt, out int maxcount, out
int[] counts, out int[] ind, out int[] val)
        {
            maxcount = 0;
            //Part 1 -> Create an array which contains the count of the input numbers
            counts = new int[M + 1];
            for (int i = 0; i < N; i++)
            {
                counts[data[i]]++;
            }
            //Part 2 -> Find maxcount, which is the the maximum occurrence a number
can have
            for (int i = 0; i < M; ++i)
            {
                if (counts[i] > maxcount)
                {
                    maxcount = counts[i];
                }
            }
            //Part 3 -> Store the values and indexes of the numbers that have occurred
“maxcount” times
            val = new int[N];
            ind = new int[N];
            cnt = 0;
            for (int i = 0; i < M; i++)
```

```
            {
                if (counts[i] == maxcount)
                {
                    val[cnt] = i;
                    for (int j = 0; j < N; j++)
                    {
                        if (i == data[j])
                        {
                            ind[cnt] = j;
                            break;
                        }
                    }
                    cnt++;
                }
            }
        }

        static void writeOutput(int cnt, int[] ind, int[] val, int maxcount)
        {
            Console.Error.WriteLine("=== Output ===");
            for (int i = 0; i < cnt; i++)
            {
                Console.WriteLine((ind[i] + 1) + " " + val[i] + " " + maxcount);
            }
        }

        static void Main(string[] args)
        {
            ///Declaration
            ///
            int N;
            int M;
            int[] data,counts, val, ind;
            int maxcount,cnt;

            ///Input
            ///
            readData(out N, out M, out data);

            ///Implementation
            ///
            solve(N, M, data, out cnt, out maxcount, out counts, out ind, out val );

            ///Output
            ///
            writeOutput(cnt, ind, val, maxcount);
        }
    }
}
```

Testing

Valid test cases

1. test case: in1.txt

Input – *5 inputs, all different*
N : 5 M:1000 $data_1$: 1 $data_2$: 2 $data_3$: 3

$data_4$: 4 $data_5$: 5
Output
1 1 1 2 2 1 3 3 1 4 4 1 5 5 1

2. test case: in2.txt

Input – *2 same numbers, only index of the first most repeated should be printed*
N : 3 M:100 $data_1$: 10 $data_2$: 10 $data_3$: 20
Output
1 10 2

3. test case: in3.txt

Input – *One input only*
N : 1 M:100 $data_1$: 43
Output
1 43 1

4. test case: in4.txt

Input – *Many inputs given, only 2 winners. Smallest number should come first*
N : 10 M:500
$data_1$: 321 $data_2$: 100 $data_3$: 50 $data_4$: 90 $data_5$: 321 $data_6$: 50 $data_7$: 50 $data_8$: 321 $data_9$: 42 $data_{10}$: 31
Output
3 50 3 1 321 3

5. test case: in5.txt

Input – *different numbers given, only one winner*
N : 8 M:1000
$data_1$: 988 $data_2$: 777 $data_3$: 888 $data_4$: 821

$data_5$: 987 $data_6$: 765 $data_7$: 565 $data_8$: 777
Output
2 777 2

Invalid test cases

6. test case

Input – *input given as string*
N : eleven M : 8
Output
Error! Wrong input for N. Please Input N & M again N=? M=?

7. test case

Input – negative number given as M
N : 11 M = -10
Output
Error! Wrong input for N. Please Input N & M again N=? M=?

…

8. test case

Input – negative number given for data
N : 5 M = 100 $data_1$: 988 $data_2$: 777 $data_3$: -1
Output
Error! Wrong input for Data. Please input data 3 again

…

9. test case

Input – string given as data
N : 5 M = 100 $data_1$: 988 $data_2$: 777 $data_3$: abs
Output
Error! Wrong input for Data. Please input data 3 again

…

Further development options

1. The program crashes if only N or M is not given at all, so an improvement can be to ask for an input again if nothing is given.
2. Identifying the errors more precisely. For example, if input is not given, it should display an error message like “No input was given.” If a string is given as an input, it should display an error message like “Wrong format of input.”
3. A friendlier interface, beyond command prompt, preferably visual.