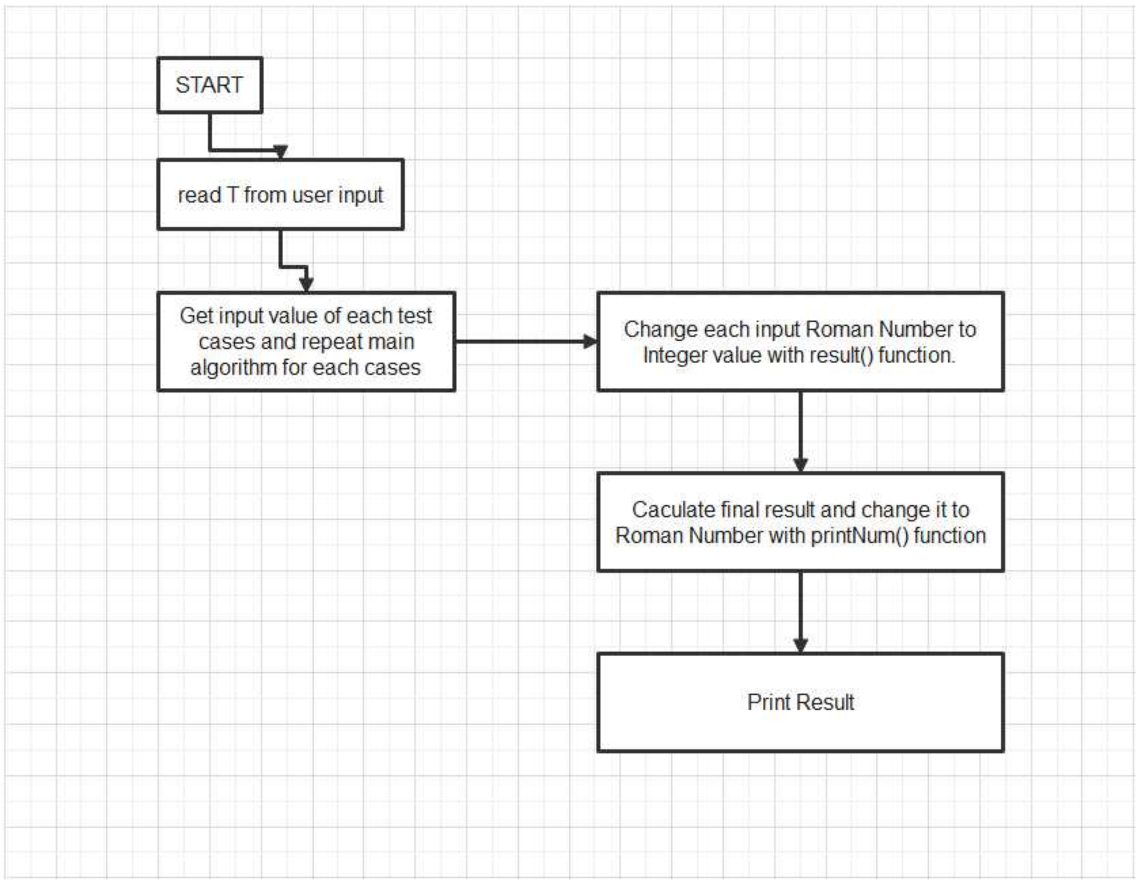
Assignment #1 : C Programming Basic

20190741 김 지수

Q1) Roman Numerals and Addition

Q1-1. Code Description with Flow chart

The code is mainly composed with the main function, change function, result function and printNum function. Main() function uses other 3 functions. Change() function changes a digit of input Roman number to its integer value. Result() function change a full Roman number to its integer value. Finally, printNum() function changes a final result which is integer value to its Roman number. The flow chart of the code is as follows.

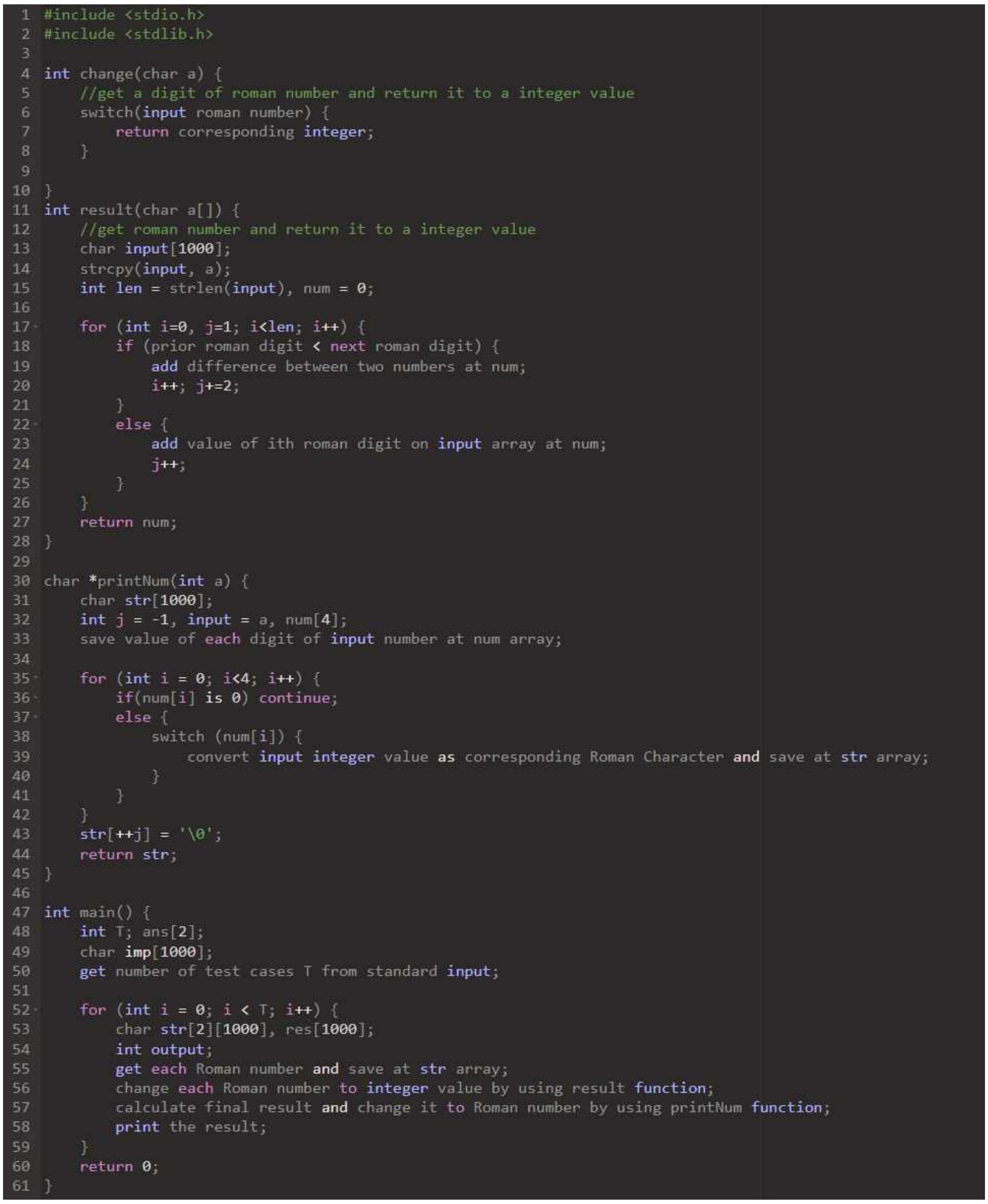

START
read T from user input
Get input value of each test cases and repeat main algorithm for each cases
Change each input Roman Number to Integer value with result() function.
Caculate final result and change it to Roman Number with printNum() function
Print Result

<Flow chart, Q1>

Q1-2. Main Algorithms with Pseudo code

The main idea for solving Q1 is using several change functions to simplify the main function that are repeated for each test cases. As it is explained at code description part, 3 functions are used for changing a digit of Roman number to Integer, a full Roman number to Integer and Integer to Roman number. By using these functions in the main function we simply get pair of Roman numbers for each test cases and changes them to calculate addition and as a result, convert final added value to Integer number in order to print the result.

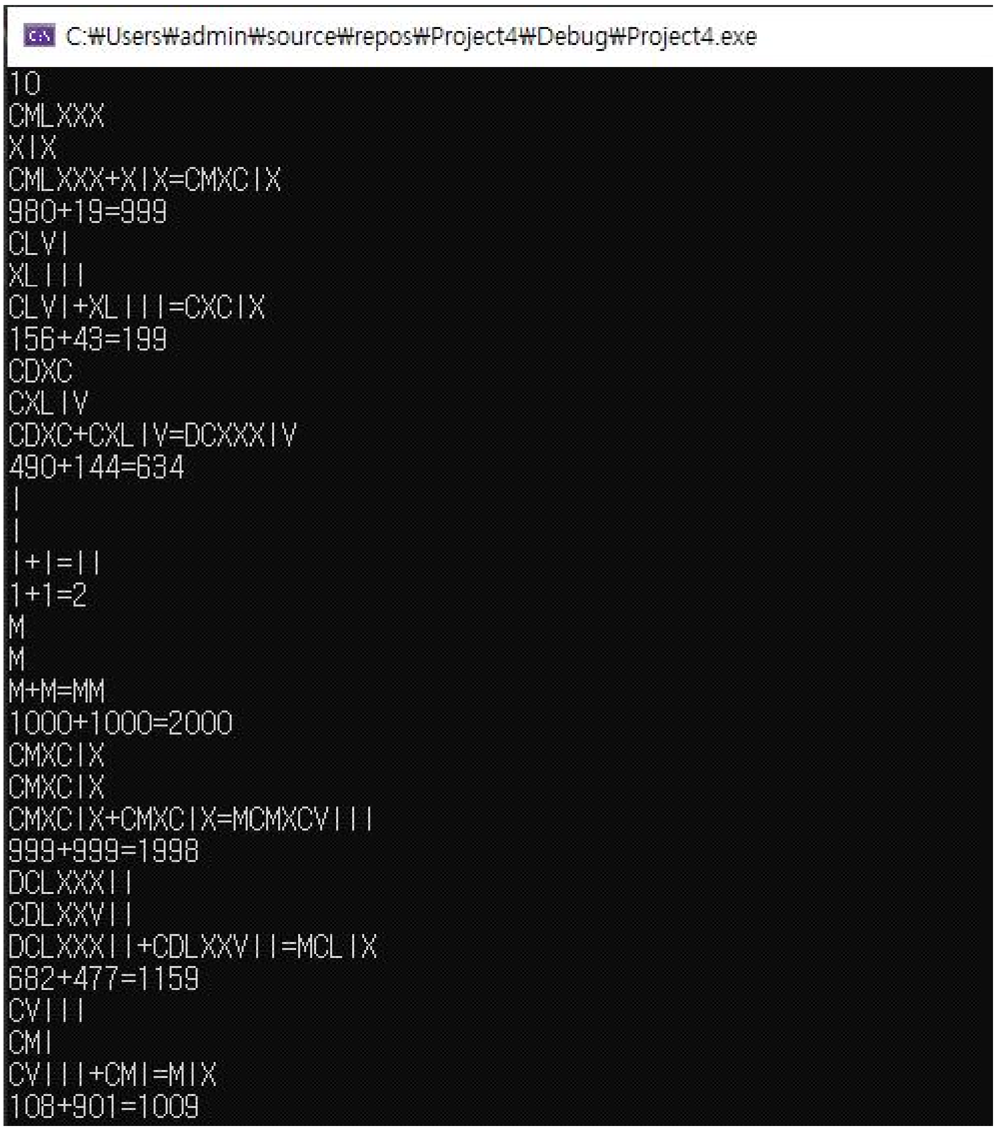
One additional idea is used at result() function. In result function, a full Roman number is converted to Integer number. At the converting process, if the prior roman digit is smaller than the next roman digit we add their differences to variable num which is the final value. Else, we add the prior roman digit at num variable and continue its process. The Pseudo code is as follows.



```
#include <stdio.h>
#include <stdlib.h>

int change(char a) {
    //get a digit of roman number and return it to a integer value
    switch(input roman number) {
        return corresponding integer;
    }

}
int result(char a[]) {
    //get roman number and return it to a integer value
    char input[1000];
    strcpy(input, a);
    int len = strlen(input), num = 0;

    for (int i=0, j=1; i<len; i++) {
        if (prior roman digit < next roman digit) {
            add difference between two numbers at num;
            i++; j+=2;
        }
        else {
            add value of ith roman digit on input array at num;
            j++;
        }
    }
    return num;
}

char *printNum(int a) {
    char str[1000];
    int j = -1, input = a, num[4];
    save value of each digit of input number at num array;

    for (int i = 0; i<4; i++) {
        if(num[i] is 0) continue;
        else {
            switch (num[i]) {
                convert input integer value as corresponding Roman Character and save at str array;
            }
        }
    }
    str[++j] = '\0';
    return str;
}

int main() {
    int T; ans[2];
    char imp[1000];
    get number of test cases T from standard input;

    for (int i = 0; i < T; i++) {
        char str[2][1000], res[1000];
        int output;
        get each Roman number and save at str array;
        change each Roman number to integer value by using result function;
        calculate final result and change it to Roman number by using printNum function;
        print the result;
    }
    return 0;
}
```

<Pseudocode, Q1>

Q1-3. Test cases simulated



```
C:₩Users₩admin₩source₩repos₩Project4₩Debug₩Project4.exe
10
CMLXXX
XIX
CMLXXX+XIX=CMXCIX
980+19=999
CLVI
XLIII
CLVI+XLIII=CXCIX
156+43=199
CDXC
CXLIV
CDXC+CXLIV=DCXXXIV
490+144=634
I
I
I+I=II
1+1=2
M
M
M+M=MM
1000+1000=2000
CMXCIX
CMXCIX
CMXCIX+CMXCIX=MCMXCVIII
999+999=1998
DCLXXXII
CDLXXVII
DCLXXXII+CDLXXVII=MCLIX
682+477=1159
CVIII
CMI
CVIII+CMI=MIX
108+901=1009
```

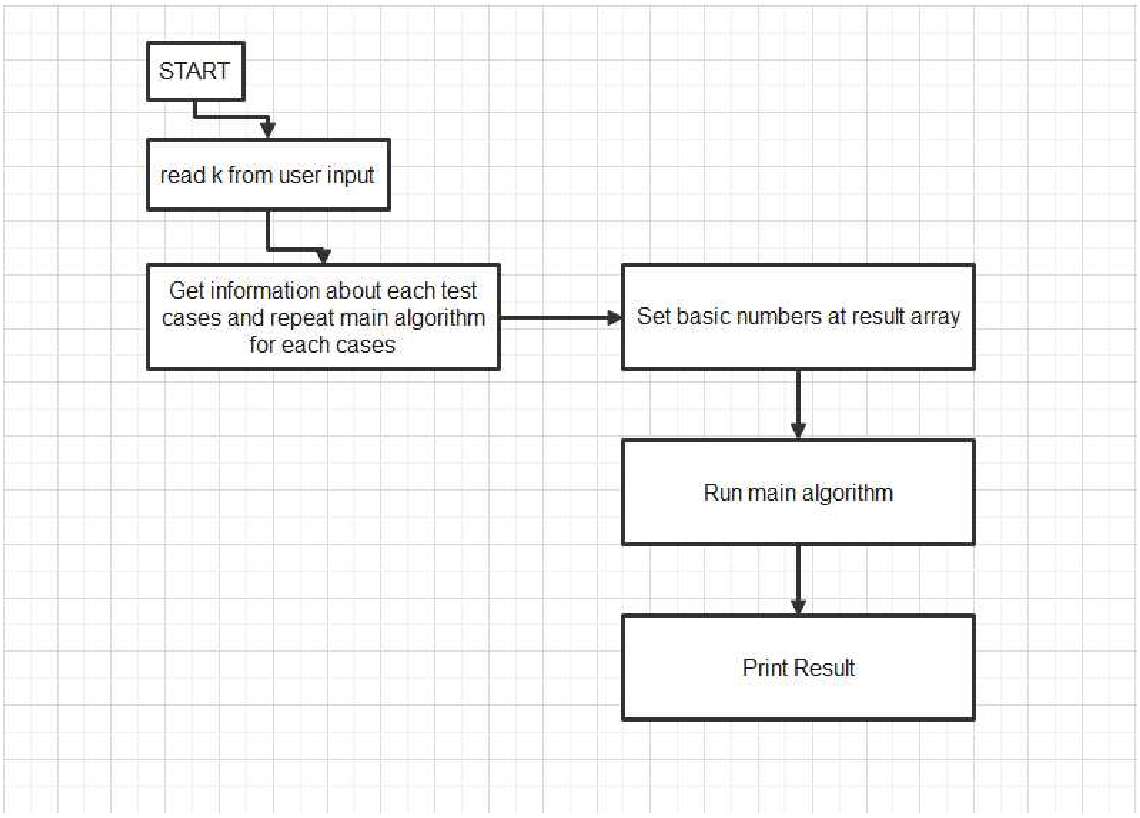
<Test results, Q1>

As simulating the random test cases could make the program more reliable by finding the c orner cases, following cases are tested. Random test cases are generated by using Roman-Inte ger converter site searched at Google. As a result, there is no program error or incorrect res ult found for several test cases.

Q2) Smart Mouse

Q2-1. Code Description with Flow chart

The code is simple to describe that it is composed only with the main function, main(). Firs t, elementary information, that is K(number of test cases given) is given and it is stored in int eger variable k by using scanf_s function. Next, FOR statement is used to repeat the main alg orithm K times to fulfill the given condition. In FOR statement row numbers M and column n umbers N for each cases are input, then it moves on to the main algorithm. The flow chart o f the code is as follows.

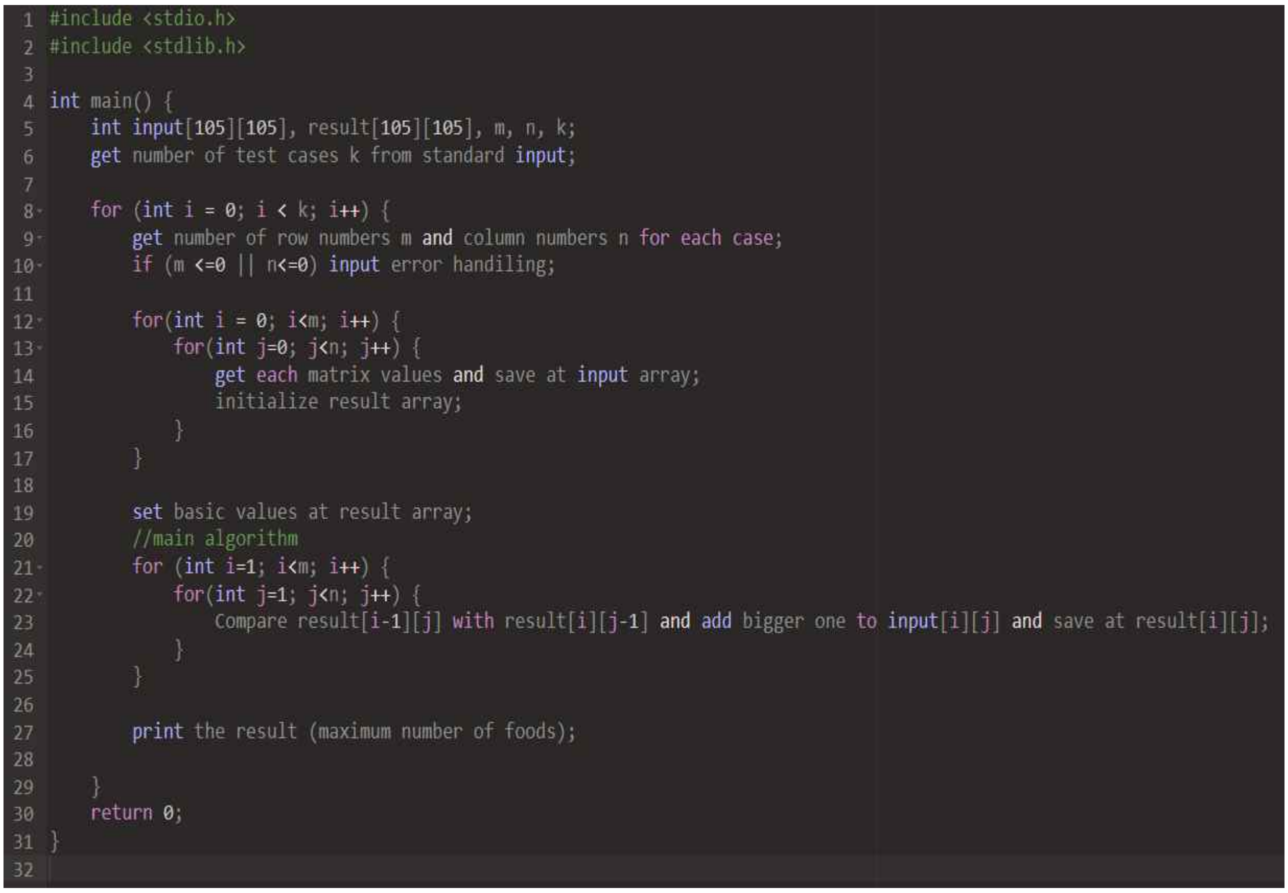

START
read k from user input
Get information about each test cases and repeat main algorithm for each cases
Set basic numbers at result array
Run main algorithm
Print Result

<Flow chart, Q2>

Q2-2. Main Algorithms with Pseudo code

The main Idea used in this problem is that by using the 2-dimensional array result[105][105] for saving and calculate the maximum number of foods that a mouse can eat at that location moment. It is easier to calculate maximum number of foods step by step in the matrix by usi ng this result array. Before starting the nested for loop each value at the first row and colum n(result[0toM-1][0] and result[0][0toN-1]is calculated first because there is only one way to rea ch result[M-1][0] or result[0][N-1] in line with the requirement that a mouse must reach the e xit with a minimum route.

Next, in the nested for loop, maximum number of foods at each position of the matrix is ca lculated by comparing result[i-1][j] with result[i][j-1]. At result[i][j], the bigger one between the two values will be added. Because the bigger value means that there are more foods on that route. After the completion of the nested for loop the final result will be printed. The Pseudo code is as follows.



```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int input[105][105], result[105][105], m, n, k;
    get number of test cases k from standard input;

    for (int i = 0; i < k; i++) {
        get number of row numbers m and column numbers n for each case;
        if (m <=0 || n<=0) input error handiling;

        for(int i = 0; i<m; i++) {
            for(int j=0; j<n; j++) {
                get each matrix values and save at input array;
                initialize result array;
            }
        }

        set basic values at result array;
        //main algorithm
        for (int i=1; i<m; i++) {
            for(int j=1; j<n; j++) {
                Compare result[i-1][j] with result[i][j-1] and add bigger one to input[i][j] and save at result[i][j];
            }
        }

        print the result (maximum number of foods);

    }
    return 0;
}

```

<Pseudocode, Q2>

Q2-3. Test cases simulated

As simulating the random test cases could make the program more reliable by finding the corner cases, following cases are tested. Random test cases are generated by M*N matrix generator that are composed of random number by using C programming language. Although the Question mentioned that the number of foods is less or equal than 100, the size of number does not affect the correctness of the program. So, the adequate size of numbers which do not make stack overflow are used in test cases and are as follows. The result of the test were checked through mutual verification with other students by sharing the test cases.

Test 1.

M = 10, N = 10

```
19 46 44  0  4 40 41 26 35 15
12 46 48  5 22  2  7 29 37  1
20 10  7 18 10 36  0 10 42 47
9 43 49 43 16  3 27 22  9 33
44  6 36 19 14 50 35 17 36  8
38  2  7  7  3  2 21 21 33 17
10 37 44  4 38 16 27  0 28 13
7 25  9 37 10 26 13 38 47 24
2 48 10 48 22 43 40 48 18  6
26 14 42 43 35 35 26 20  3 33
```

Result : 606

Test 2.

M = 4 , N = 28

```
19 46 44  0  4 40 41 26 35 15 12 46 48  5 22  2  7 29 37  1 20 10  7 18 10 36  0 10
42 47  9 43 49 43 16  3 27 22  9 33 44  6 36 19 14 50 35 17 36  8 38  2  7  7  3  2
21 33 17 10 37 44  4 38 16 27  0 28 13  7 25  9 37 10 26 13 38 47 24  2 48 10 48 22
43 40  7  3 13 38 35 49 19 12 26 19 22  6 38 16 39  1 46  3 17 34 34 11 23  9 12  1
```

Result : 873