The CENTRE for EDUCATION
in MATHEMATICS and COMPUTING

2012 Canadian Computing Competition: Senior Division

Sponsor:

University of
Waterloo

Canadian Computing Competition
Student Instructions for the Senior Problems

1. You may only compete in one competition. If you wish to write the Junior paper, see the other problem set.

2. Be sure to indicate on your **Student Information Form** that you are competing in the **Senior** competition.

3. You have three (3) hours to complete this competition.

4. You should assume that

   - all input is from a file named `sX.in`, where `X` is the problem number ($1 \leq \texttt{X} \leq 5$).
   - all output is to the screen

   Since your program will read from a file, there is no need for prompting. Be sure your output matches the expected output in terms of order, spacing, etc. IT MUST MATCH EXACTLY!

5. Do your own work. Cheating will be dealt with harshly.

6. Do not use any features that the judge (your teacher) will not be able to use while evaluating your programs.

7. Books and written materials are allowed. Any machine-readable materials (like other programs which you have written) are *not* allowed. However, you are allowed to use "standard" libraries for your programming languages; for example, the STL for C++, java.util.*, java.io.*, etc. for Java, and so on.

8. Applications other than editors, compilers, debuggers or other standard programming tools are **not** allowed. Any use of other applications will lead to disqualification.

9. Please use file names that are unique to each problem: for example, please use `s1.pas` or `s1.c` or `s1.java` (or some other appropriate extension) for Problem S1. This will make the evaluator's task a little easier.

10. Your program will be run against test cases other than the sample ones. Be sure you test your program on other test cases. Inefficient solutions may lose marks for some problems, especially Problems 4 and 5. Be sure your code is as efficient (in terms of time) as possible.

11. Note that the top 2 Senior competitors in each region of the country will get a plaque and $100, and the schools of these competitors will also get a plaque. The regions are:

    - West (BC to Manitoba)
    - Ontario North and East
    - Metro Toronto area

- Ontario Central and West
- Quebec and Atlantic

12. If you finish in the top 20 competitors on this competition, you will be invited to participate in CCC Stage 2, held at the University of Waterloo in May 2012. We will select the Canadian International Olympiad in Informatics (IOI) team from among the top contestants at Stage 2. You should note that IOI 2012 will be held in Italy. Note that you will need to know C, C++ or Pascal if you are invited to Stage 2. But first, do well on this contest!

13. Check the CCC website at the end of March to see how you did on this contest, and to see who the prize winners are. The CCC website is:

`www.cemc.uwaterloo.ca/ccc`

Problem S1: Don't pass me the ball!

Problem Description

A CCC soccer game operates under slightly different soccer rules. A goal is only counted if the 4 players, in order, who touched the ball prior to the goal have jersey numbers that are in strictly increasing numeric order with the highest number being the goal scorer.

Players have jerseys numbered from 1 to 99 (and each jersey number is worn by exactly one player).

Given a jersey number of the goal scorer, indicate how many possible combinations of players can produce a valid goal.

Input Specification

The input will be the positive integer J ($1 \leq J \leq 99$), which is the jersey number of the goal scorer.

Output Specification

The output will be one line containing the number of possible scoring combinations that could have J as the goal scoring jersey number.

Sample Input 1

```
4
```

Output for Sample Input 1

```
1
```

Sample Input 2

```
2
```

Output for Sample Input 2

```
0
```

Sample Input 3

```
90
```

Output for Sample Input 3

```
113564
```

Problem S2: Aromatic Numbers

Problem Description

This question involves calculating the value of *aromatic* numbers which are a combination of Arabic digits and Roman numerals.

An aromatic number is of the form $ARARAR \ldots AR$, where each A is an Arabic digit, and each R is a Roman numeral. Each pair AR contributes a value described below, and by adding or subtracting these values together we get the value of the entire aromatic number.

An Arabic digit A can be 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9. A Roman numeral R is one of the seven letters I, V, X, L, C, D, or M. Each Roman numeral has a base value:

Symbol	I	V	X	L	C	D	M
Base value	1	5	10	50	100	500	1000

The value of a pair AR is A times the base value of R. Normally, you add up the values of the pairs to get the overall value. However, wherever there are consecutive symbols $ARA'R'$ with R' having a *strictly bigger* base value than R, the value of pair AR must be *subtracted* from the total, instead of being *added*.

For example, the number 3M1D2C has the value $3 * 1000 + 1 * 500 + 2 * 100 = 3700$ and 3X2I4X has the value $3 * 10 - 2 * 1 + 4 * 10 = 68$.

Write a program that computes the values of aromatic numbers.

Input Specification
The input is a valid aromatic number consisting of between 2 and 20 symbols.

Output Specification
The output is the decimal value of the given aromatic number.

Sample Input 1
```
3M1D2C
```

Output for Sample Input 1
```
3700
```

Sample Input 2
```
2I3I2X9V1X
```

Output for Sample Input 2
```
-16
```

Problem S3: Absolutely Acidic

Problem Description

You are gathering readings of acidity level in a very long river in order to determine the health of the river. You have placed N sensors ($2 \leq N \leq 2\ 000\ 000$) in the river, and each sensor gives an integer reading R ($1 \leq R \leq 1\ 000$). For the purposes of your research, you would like to know the frequency of each reading, and find the absolute difference between the two most frequent readings.

If there are more than two readings that have the highest frequency, the difference computed should be the *largest* such absolute difference between two readings with this frequency. If there is only one reading with the largest frequency, but more than one reading with the second largest frequency, the difference computed should be the *largest* absolute difference between the most frequently occurring reading and any of the readings which occur with second-highest frequency.

Input Specification

The first line of input will be the integer N ($2 \leq N \leq 2\ 000\ 000$), the number of sensors. The next N lines each contain the reading for that sensor, which is an integer R ($1 \leq R \leq 1\ 000$). You should assume that there are at least two different readings in the input.

Output Specification

Output the positive integer value representing the absolute difference between the two most frequently occurring readings, subject to the tie-breaking rules outlined above.

Sample Input 1

```
5
1
1
1
4
3
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
4
10
6
1
8
```

Output for Sample Input 2

```
9
```

Problem S4: A Coin Game

Problem Description

When she is bored, Jo Coder likes to play the following game with coins on a table. She takes a set of distinct coins and lines them up in a row. For example, let us say that she has a penny (P, worth \$0.01), a nickel (N, worth \$0.05), and a dime (D, worth \$0.10). She lines them up in an arbitrary order, (for example, D N P), and then moves them around with the goal of placing them in strictly increasing order by value, that is P N D (i.e., \$0.01, \$0.05, \$0.10). She has particular rules that she follows:

- The initial coin line-up defines all positions where coins can be placed. That is, no additional positions can be added later, and even if one of the positions does not have a coin on it at some point, the position still exists.
- The game consists of a sequence of moves and in each move Jo moves a coin from one position to an *adjacent* position.
- The coins can be stacked, and in a move Jo always takes the top coin from one stack and moves it to the top of another stack.
- In a stack of coins, Jo never places a higher-value coin on top of a lower-value coin.

For simplicity, let the coins have consecutive integer values (e.g., denote the penny as 1, nickel as 2, and dime as 3). Then, in the above example, Jo could play the game in the following way in 20 moves (where XY denotes that coin X is on top of coin Y):

Move	Position 1	Position 2	Position 3
initial	3	2	1
1	3	12	
2	13	2	
3	13		2
4	3	1	2
5	3		12
6		3	12
7		13	2
8	1	3	2
9	1	23	
10		123	

Move	Position 1	Position 2	Position 3
11		23	1
12	2	3	1
13	2	13	
14	12	3	
15	12		3
16	2	1	3
17	2		13
18		2	13
19		12	3
20	1	2	3

For some starting configurations, it is not always possible to obtain the goal of strictly increasing order.

Input Specification

The input will contain some number of test cases. A test case consists of two lines. The first line contains a positive integer n ($n < 8$), which is the number of coins. We assume that the coins are labeled 1, 2, 3, ... n. The second line contains a list of numbers 1 to n in an arbitrary order, which represents the initial coin configuration. For the above example, the input test case would be:

```
3
3 2 1
```

The end of test cases is indicated by 0 on a line by itself.

Output Specification

For each test case, output one line, which will either contain the *minimal number* of moves in which Jo can achieve the goal coin line-up, or, if it is not possible to achieve the goal coin line-up, `IMPOSSIBLE`.

Sample Input

```
3
3 2 1
2
2 1
0
```

Output for Sample Input

```
20
IMPOSSIBLE
```

Problem S5: Mouse Journey

Problem Description

You are a mouse that lives in a cage in a large laboratory. The laboratory is composed of one rectangular grid of square cages, with a total of R rows and C columns of cages ($1 \leq R, C \leq 25$).

To get your exercise, the laboratory owners allow you to move between cages. You can move between cages either by moving right between two adjacent cages in the same row, or by moving down between two adjacent cages in the same column. You cannot move diagonally, left or up.

Your cage is in one corner of the laboratory, which has the label $(1, 1)$ (to indicate top-most row, left-most column). You would like to visit your brother who lives in the cage labelled (R, C) (bottom-most row, right-most column), which is in the other corner diagonally. However, there are some cages which you cannot pass through, since they contain cats.

Your brother, who loves numbers, would like to know how many different paths there are between your cage and his that do not pass through any cat cage. Write a program to compute this number of cat-free paths.

Input Specification

The first line of input contains two integers R and C, separated by one space representing the number of rows and columns (respectively). On the second line of input is the integer K, the number of cages that contain cats. The next K lines each contain the row and column positions (in that order) for a cage that contains a cat. None of the K cat cages are repeated, and all cages are valid positions. Note also that $(1, 1)$ and (R, C) will not be cat cages.

Output Specification

Output the non-negative integer value representing the number of paths between your cage at position $(1, 1)$ and your brother's cage at position (R, C). You can assume the output will be strictly less than $1\ 000\ 000\ 000$.

Sample Input 1

```
2 3
1
2 1
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
3 4
3
2 3
2 1
```

```
1 4
```

Output for Sample Input 2

```
1
```