The CENTRE for EDUCATION
in MATHEMATICS and COMPUTING

2012 Canadian Computing Competition: Junior Division

Sponsor:

University of
Waterloo

Canadian Computing Competition
Student Instructions for the Junior Problems

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

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

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

4. You should assume that

   - all input is from the keyboard
   - all output is to the screen

   For some problems, you may be asked for prompting: please provide this for the user. If no prompting is required, you do not need to provide any. 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 `j1.pas` or `j1.c` or `j1.java` (or some other appropriate extension) for Problem J1. 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. Be sure your code is as efficient (in terms of time) as possible.

11. Note that the top 2 Junior 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. 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 J1: Speed fines are not fine!

Problem Description
Many communities now have "radar" signs that tell drivers what their speed is, in the hope that they will slow down.

You will output a message for a "radar" sign. The message will display information to a driver based on his/her speed according to the following table:

km/h over the limit	Fine
1 to 20	$100
21 to 30	$270
31 or above	$500

Input Specification
The user will be prompted to enter two integers. First, the user will be prompted to enter the speed limit. Second, the user will be prompted to enter the recorded speed of the car.

Output Specification
If the driver is not speeding, the output should be:

`Congratulations, you are within the speed limit!`

If the driver is speeding, the output should be:

`You are speeding and your fine is $`F.

where F is the amount of the fine as described in the table above.

Sample Session 1 (with output shown in `text`, user input in *italics*)
`Enter the speed limit:` *40*
`Enter the recorded speed of the car:` *39*
`Congratulations, you are within the speed limit!`

Sample Session 2
`Enter the speed limit:` *100*
`Enter the recorded speed of the car:` *131*
`You are speeding and your fine is $500.`

Sample Session 3
`Enter the speed limit:` *100*
`Enter the recorded speed of the car:` *120*
`You are speeding and your fine is $100.`

Problem J2: Sounds fishy!

Problem Description
A fish-finder is a device used by anglers to find fish in a lake. If the fish-finder finds a fish, it will sound an alarm. It uses depth readings to determine whether to sound an alarm. For our purposes, the fish-finder will decide that a fish is swimming past if:

- there are four consecutive depth readings which form a strictly increasing sequence (such as 3 4 7 9) (which we will call "Fish Rising"), or
- there are four consecutive depth readings which form a strictly decreasing sequence (such as 9 6 5 2) (which we will call "Fish Diving"), or
- there are four consecutive depth readings which are identical (which we will call "Constant Depth").

All other readings will be considered random noise or debris, which we will call "No Fish."

Your task is to read a sequence of depth readings and determine if the alarm will sound.

Input Specification
The input will be four positive integers, representing the depth readings. Each integer will be on its own line of input.

Output Specification
The output is one of four possibilities. If the depth readings are increasing, then the output should be `Fish Rising`. If the depth readings are decreasing, then the output should be `Fish Diving`. If the depth readings are identical, then the output should be `Fish At Constant Depth`. Otherwise, the output should be `No Fish`.

Sample Input 1
```
30
10
20
20
```

Output for Sample Input 1
```
No Fish
```

Sample Input 2
```
1
10
12
13
```

Output for Sample Input 2
```
Fish Rising
```

Problem J3: Icon Scaling

Problem Description
You have been asked to take a small icon that appears on the screen of a smart telephone and scale it up so it looks bigger on a regular computer screen.

The icon will be encoded as characters (`x` and `*`) in a 3×3 grid as follows:

```
*x*
 xx
*  *
```

Write a program that accepts a positive integer scaling factor and outputs the scaled icon. A scaling factor of k means that each character is replaced by a $k \times k$ grid consisting only of that character.

Input Specification
The input will be a positive integer k such that $k < 25$.

Output Specification
The output will be $3k$ lines, which represent each individual line scaled by a factor of k and repeated k times. A line is scaled by a factor of k by replacing each character in the line with k copies of the character.

Sample Input
```
3
```

Output for Sample Input
```
***xxx***
***xxx***
***xxx***
   xxxxxx
   xxxxxx
   xxxxxx
***   ***
***   ***
***   ***
```

Problem J4: Big Bang Secrets

Problem Description
Sheldon and Leonard are physicists who are fixated on the BIG BANG theory. In order to exchange secret insights they have devised a code that encodes UPPERCASE words by shifting their letters forward.

Shifting a letter by S positions means to go forward S letters in the alphabet. For example, shifting B by $S = 3$ positions gives E. However, sometimes this makes us go past Z, the last letter of the alphabet. Whenever this happens we wrap around, treating A as the letter that follows Z. For example, shifting Z by $S = 2$ positions gives B.

Sheldon and Leonard's code depends on a parameter K and also varies depending on the position of each letter in the word. For the letter at position P, they use the shift value of $S = 3P + K$.

For example, here is how ZOOM is encoded when $K = 3$. The first letter Z has a shift value of $S = 3 \times 1 + 3 = 6$; it wraps around and becomes the letter F. The second letter, O, has $S = 3 \times 2 + 3 = 9$ and becomes X. The last two letters become A and B. So Sheldon sends Leonard the secret message: FXAB

Write a program for Leonard that will **decode** messages sent by Sheldon.

Input Specification
The input will be two lines. The first line will contain the positive integer K ($K < 10$), which is used to compute the shift value. The second line of input will be the word, which will be a sequence of uppercase characters of length at most 20.

Output Specification
The output will be the decoded word of uppercase letters.

Sample Input 1
```
3
FXAB
```

Output for Sample Input 1
```
ZOOM
```

Sample Input 2
```
5
JTUSUKG
```

Output for Sample Input 2
```
BIGBANG
```

Problem J5: 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 ($n < 5$), 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
```