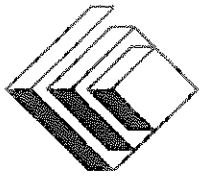

The CENTRE for EDUCATION
in MATHEMATICS and COMPUTING

# *2015 Canadian Computing Competition: Junior Division*

Sponsor:

WATERLOO
MATHEMATICS

# *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.

   - if your supervising teacher is grading your solutions, all input is from the keyboard;
   - if you are using the On-line CCC grader, all input is from standard input;
   - all output is to standard output (i.e., to the screen).

   There is no need for prompting. Be sure your output matches the expected output in terms of order, spacing, etc. IT MUST MATCH EXACTLY!

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

5. Do not use any features that the judge (your teacher or the On-line Grader) will not be able to use while evaluating your programs. In particular, take note of the type and version of the compiler used for your programming language on the On-line Grader if you are using the On-line Grader.

6. 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.

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

8. If your teacher is grading, please use file names that are unique to each problem: use `j1.pas` or `j1.c` or `j1.java` (or some other appropriate extension) for Problem J1. If you are using the On-line Grader, follow naming rules described there (and take particular note of file and class names for Java programs).

9. 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. You will have at most 5 seconds of execution time per test case.

10. 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: Special Day

**Problem Description**
February 18 is a special date for the CCC this year.

Write a program that asks the user for a numerical month and numerical day of the month and then determines whether that date occurs before, after, or on February 18.

If the date occurs before February 18, output the word `Before`. If the date occurs after February 18, output the word `After`. If the date is February 18, output the word `Special`.

**Input Specification**
The input consists of two integers each on a separate line. These integers represent a date in 2015.

The first line will contain the month, which will be an integer in the range from 1 (indicating January) to 12 (indicating December).

The second line will contain the day of the month, which will be an integer in the range from 1 to 31. You can assume that the day of the month will be valid for the given month.

**Output Specification**
Exactly one of `Before`, `After` or `Special` will be printed on one line.

**Sample Input 1**
```
1
7
```

**Output for Sample Input 1**
```
Before
```

**Sample Input 2**
```
8
31
```

**Output for Sample Input 2**
```
After
```

**Sample Input 3**
```
2
18
```

**Output for Sample Input 3**
```
Special
```

# Problem J2: Happy or Sad

**Problem Description**
We often include emoticons in our text messages to indicate how we are feeling. The three consecutive characters `:-)` indicate a happy face and the three consecutive characters `:-(` indicate a sad face. Write a program to determine the overall mood of a message.

**Input Specification**
There will be one line of input that contains between 1 and 255 characters.

**Output Specification**
The output is determined by the following rules:

- If the input line does not contain any happy or sad emoticons, output `none`.
- Otherwise, if the input line contains an equal number of happy and sad emoticons, output `unsure`.
- Otherwise, if the input line contains more happy than sad emoticons, output `happy`.
- Otherwise, if the input line contains more sad than happy emoticons, output `sad`.

**Sample Input 1**
```
How are you :-) doing :-( today :-)?
```

**Output for Sample Input 1**
```
happy
```

**Sample Input 2**
```
:)
```

**Output for Sample Input 2**
```
none
```

**Sample Input 3**
```
This:-(is str:-(:-(ange te:-)xt.
```

**Output for Sample Input 3**
```
sad
```

# Problem J3: Rövarspråket

**Problem Description**
In Sweden, there is a simple child's game similar to Pig Latin called Rövarspråket (Robbers Language).

In the CCC version of Rövarspråket, every consonant is replaced by three letters, in the following order:

- the consonant itself;
- the vowel closest to the consonant in the alphabet (e.g., if the consonant is d, then the closest vowel is e), with the rule that if the consonant falls exactly between two vowels, then the vowel closer to the start of the alphabet will be chosen (e.g., if the consonant is c, then the closest vowel is a);
- the next consonant in the alphabet following the original consonant (e.g., if the consonant is d, then the next consonant is f) except if the original consonant is z, in which case the next consonant is z as well.

Vowels in the word remain the same. (Vowels are a, e, i, o, u and all other letters are consonants.)

Write a program that translates a word from English into Rövarspråket.

**Input Specification**
The input consists of one word entirely composed of lower-case letters. There will be at least one letter and no more than 30 letters in this word.

**Output Specification**
Output the word as it would be translated into Rövarspråket on one line.

**Sample Input 1**
```
joy
```

**Output for Sample Input 1**
```
jikoyuz
```

**Sample Input 2**
```
ham
```

**Output for Sample Input 2**
```
hijamon
```

# Problem J4: Wait Time

**Problem Description**
You exchange text messages with your friends. Since you receive so many messages, you want to measure how long your friends have to wait for your replies.

Your message device records each received and sent message in order using the following two kinds of entries:

- `R X` indicates a message was received from a friend numbered $X$;
- `S X` indicates a message was sent to a friend numbered $X$.

Your message device sends and receives messages instantaneously, and for each consecutive pair of entries described above, either

- a single entry `W X` is recorded in between them indicating they occur $X$ seconds apart, or
- there is no entry between them and they occur one second apart.

Several rules of message etiquette are always followed:

- the only messages you send are replies to messages that you have received;
- you send at most one reply to any message from any particular friend;
- your friends do not send a subsequent message until you have replied to their previous message.

The wait time for a message is the time that passes between when you receive it and the time you reply to it. If a friend $X$ received a reply to each message they sent, the total wait time for friend $X$ is the sum of all wait times for all messages from friend $X$. Otherwise, the total wait time for friend $X$ is $-1$.

Your job is to determine the total wait time for each friend.

**Input Specification**
The input consists of the integer $M$ ($1 \leq M \leq 20$), followed by $M$ lines, where each line consists of one character (`W`, `R`, or `S`), followed by a space, followed by an integer $X$ ($1 \leq X \leq 100$). These $M$ lines are the entries described above (in order).

**Output Specification**
Output one line for each friend that sent a message in the form $X$ $T$ where $X$ is a friend number and $T$ is the total wait time for friend $X$. The lines are in increasing order of the friend numbers.

**Sample Input 1**

```
5
R 2
R 3
W 5
S 2
S 3
```

**Output for Sample Input 1**

```
2 6
3 6
```

**Explanation of Output for Sample Input 1**

Friend 2 sends a message at time 0 and Friend 3 sends a message at time 1. Friend 2 receives a reply at time 6 and Friend 3 receives a reply at time 7.

**Sample Input 2**

```
14
R 12
W 2
R 23
W 3
R 45
S 45
R 45
S 23
R 23
W 2
S 23
R 34
S 12
S 34
```

**Output for Sample Input 2**

```
12 13
23 8
34 2
45 -1
```

**Explanation of Output for Sample Input 2**

For Friend 12, a message is received at time 0 and replied to at time 13. For Friend 23, two messages are exchanged, with the first message having a wait time of 6 seconds and the second message having a wait time of 2 seconds. For Friend 34, a message is received at time 10 and replied to at time 12. Friend 45 sends a message which is never replied to.

# Problem J5: $\pi$-day

**Problem Description**

You may know that March 14 is known as "$\pi$-day", since 3.14 (which is the third month and fourteenth day) is a good approximation of $\pi$.

Mathematicians celebrate this day by eating pie.

Suppose that you have $n$ pieces of pie, and $k$ people who are lined up for pieces of pie. All $n$ pieces of pie will be given out. Each person will get at least one piece of pie, but mathematicians are a bit greedy at times. So, they always get at least as many of pieces of pie as the person in front of them.

For example, if you have 8 pieces of pie and 4 people in line, you could give out pieces of pie in the following five ways (with the first person in line being the first number in the list): $[1, 1, 1, 5]$, $[1, 1, 2, 4]$, $[1, 1, 3, 3]$, $[1, 2, 2, 3]$, $[2, 2, 2, 2]$.

Notice that if $k = n$, there is only one way to give out the pieces of pie: every person gets exactly one piece. Also, if $k = 1$, there is only one way to give out the pieces of pie: that single person gets all the pieces.

Write a program that determines the number of ways that the pieces of pie can be given out.

**Input Specification**

The first line of input is the integer number of pieces of pie, $n$ $(1 \leq n \leq 250)$.
The second line of input is the integer $k$ which is the number of people in line $(1 \leq k \leq n)$.

For at least 20% of the marks for this problem, $n \leq 9$. For at least 50% of the marks for this problem, $n \leq 70$. For at least 85% of the marks for this problem, $n \leq 120$.

**Output Specification**

The output will consist of a single integer which is the number of ways that the pieces of pie can be distributed. The output is guaranteed to be less that $2^{31}$.

**Sample Input 1**

```
8
4
```

**Output for Sample Input 1**

```
5
```

**Sample Input 2**

```
6
2
```

**Output for Sample Input 2**

```
3
```