The CENTRE for EDUCATION
in MATHEMATICS and COMPUTING

2009 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 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, how the problems were meant to be solved, and to see who the prize winners are. The CCC website is:

`www.cemc.uwaterloo.ca/ccc`

Problem J1: ISBN

Problem Description
The International Standard Book Number (ISBN) is a 13-digit code for identifying books. These numbers have a special property for detecting whether the number was written correctly.

The 1-3-sum of a 13-digit number is calculated by multiplying the digits alternately by 1's and 3's (see example) and then adding the results. For example, to compute the 1-3-sum of the number 9780921418948 we add

$$9*1+7*3+8*1+0*3+9*1+2*3+1*1+4*3+1*1+8*3+9*1+4*3+8*1$$

to get 120.

The special property of an ISBN number is that its 1-3-sum is always a multiple of 10.

Write a program to compute the 1-3-sum of a 13-digit number. To reduce the amount of typing, you may assume that the first ten digits will always be 9780921418, like the example above. Your program should input the last three digits and then print its 1-3-sum. Use a format similar to the samples below.

Sample Interactive Session 1 (user input shown in *italics*)
```
Digit 11? 9
Digit 12? 4
Digit 13? 8
```

Output for Sample Interactive Session 1
```
The 1-3-sum is 120
```

Sample Interactive Session 2 (user input shown in *italics*)
```
Digit 11? 0
Digit 12? 5
Digit 13? 2
```

Output for Sample Interactive Session 2
```
The 1-3-sum is 108
```

Problem J2: Old Fishin' Hole

Problem Description
Fishing habitat and fish species are a resource that must be carefully managed to ensure that they will be there for the future. Accordingly, fishing limits have been established for a particular river based on the population of each species. Specifically, *points* are associated with the fish caught and the total points you catch must be less than or equal to the points allowed for that river.

As an example, suppose each brown trout counts as 2 points, each northern pike counts as 5 points and each yellow pickerel counts as 2 points, and the total points allowed must be less than or equal to 12. One acceptable catch could consist of 3 brown trout and 1 northern pike, but, other combinations would also be allowed.

Your job is to write a program to input the points allocated for a river, and find how many different ways an angler who catches *at least* one fish can stay within his/her limit.

Input Description
You will be given 4 integers, one per line, representing trout points, pike points, pickerel points, and total points allowed in that order.

You can assume that each integer will be greater than 0 and less than or equal to 100.

Output Description
For each different combination of fish caught, output the combination of brown trout, northern pike, and yellow pickerel in that order. The combinations may be listed in any order. The last line of output should display the total number of unique ways to catch fish within the established limit.

Sample Input
```
1
2
3
2
```

Sample Output
```
1 Brown Trout, 0 Northern Pike, 0 Yellow Pickerel
2 Brown Trout, 0 Northern Pike, 0 Yellow Pickerel
0 Brown Trout, 1 Northern Pike, 0 Yellow Pickerel
Number of ways to catch fish: 3
```

Problem J3: Good times

Problem Description
A mobile cell service provider in Ottawa broadcasts an automated time standard to its mobile users that reflects the local time at the user's actual location in Canada. This ensures that text messages have a valid local time attached to them.

For example, when it is 1420 in Ottawa on Tuesday February 24, 2009 (specified using military, 24 hour format), the times across the country are shown in the table below:

Pacific Time	Mountain Time	Central Time	Eastern Time	Atlantic Time	Newfoundland Time
Victoria, BC Tuesday 2/24/2009 1120 PST	Edmonton, AB Tuesday 2/24/2009 1220 MST	Winnipeg, MB Tuesday 2/24/2009 1320 CST	Toronto, ON Tuesday 2/24/2009 1420 EST	Halifax, NS Tuesday 2/24/2009 1520 AST	St. John's, NL Tuesday 2/24/2009 1550 Newfoundland ST

Write a program that accepts the time in Ottawa in 24 hour format and outputs the local time in each of the cities listed above including Ottawa. You should assume that the input time will be valid (i.e., an integer between 0 and 2359 with the last two digits being between 00 and 59).

You should note that 2359 is one minute to midnight, midnight is 0, and 13 minutes after midnight is 13. You do not need to print leading zeros, and input will not contain any extra leading zeros.

Sample Input
```
1300
```

Sample Output
```
1300 in Ottawa
1000 in Victoria
1100 in Edmonton
1200 in Winnipeg
1300 in Toronto
1400 in Halifax
1430 in St. John's
```

Problem J4: Signage

Problem Description
The student council at Central Canada Collegiate is preparing signs with the message `WELCOME TO CCC GOOD LUCK TODAY` on various walls around the school. A sign is wide enough to hold w characters on each row, including spaces, as befits the wall to be decorated.

Here is how the words are put onto a sign. First, as many words as possible are placed on the first line, without exceeding the w character limit. The first word in the line begins in the leftmost position. If there is more than one word in the line, the last word ends in the rightmost position. Extra spaces are inserted into the gaps between the words so that the gaps are as similar as possible. If the gaps cannot be made equal, all of the larger gaps should appear to the left of the smaller ones. Subsequent lines are constructed in the same way.

Your program will read the available width w and output the sign on the screen. Use the "`.`" character to indicate a space.

Constraints
You may assume that $w \geq 7$.

Sample Input 1 (user inputs are shown in *italics*)
`Enter w:` *15*

Output for Sample Input 1
```
WELCOME..TO.CCC
GOOD.LUCK.TODAY
```

Sample Input 2 (user inputs are shown in *italics*)
`Enter w:` *26*

Output for Sample Input 2
```
WELCOME..TO..CCC.GOOD.LUCK
TODAY......................
```

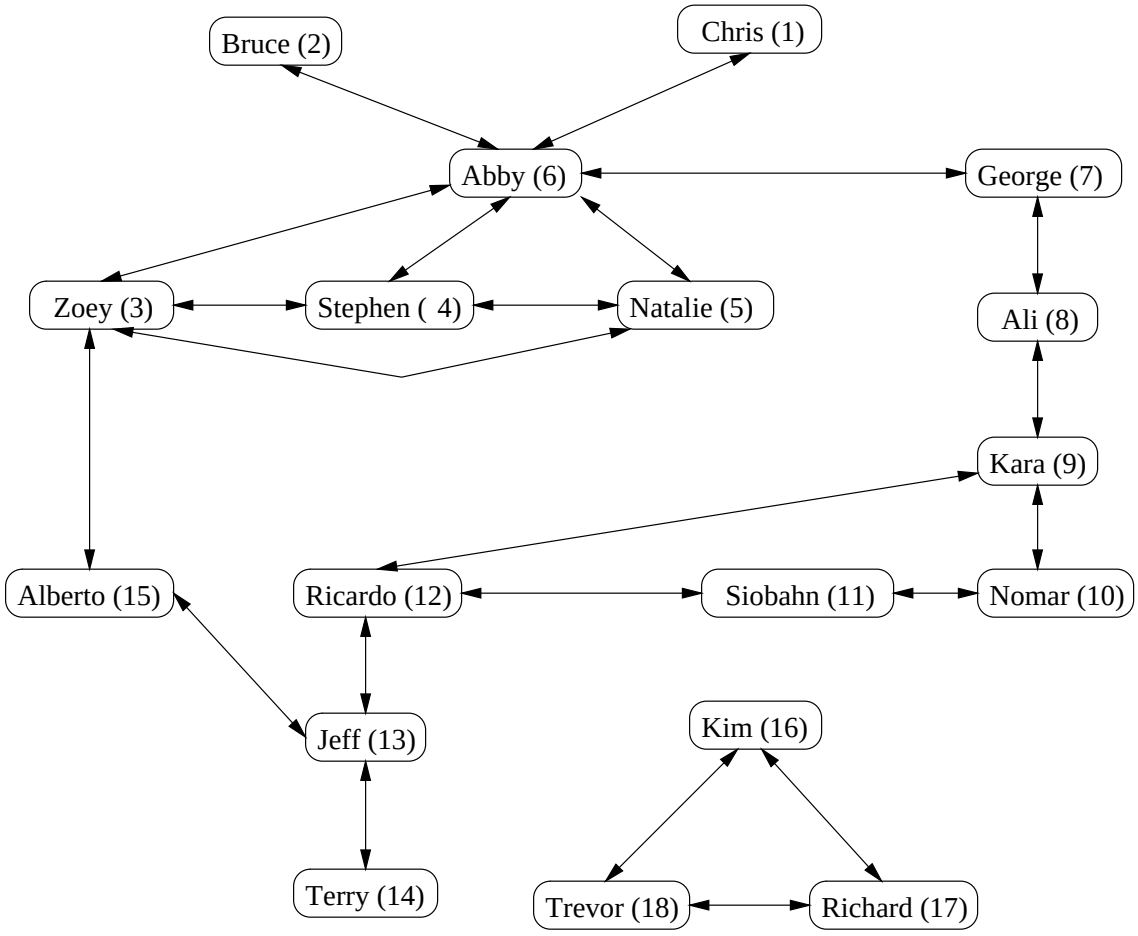
Problem J5: Degrees of Separation

Problem Description
The main socializing tool for students today is Facebook. There are many interesting computational questions connected to Facebook, such as the "degree of separation" between two people.

For example, in the diagram below, there are many different paths between Abby and Alberto. Some of these paths are:

- Abby → Zoey → Alberto
- Abby → Natalie → Zoey → Alberto
- Abby → George → Ali → Kara → Richardo → Jeff → Alberto

The shortest path between Abby and Alberto has two steps (Abby → Zoey, and Zoey → Alberto), so we say the degree of separation is 2. Additionally, Alberto would be a friend of a friend of Abby.



You can assume an initial configuration of who is friends with who as outlined in the diagram above. You will need to store these relationships in your program. These relationships can change though, and your program needs to handle these changes. In particular, friendships can begin, possibly with new people. Friendships can end. You should be able to find friends of friends and determine the degree of separation between two people.

Input/Output Description
Your program will read in six possible commands, with the action to be performed by your program outlined below. You may assume that x and y are integers, with $x \neq y$, $x \geq 1$, $y \geq 1$, $x < 50$ and $y < 50$. You may also assume that instructions (`i`, `d`, `n`, `f`, `s`, `q`) occur one per line and parameters (zero, one or two integers) occur one per line.

- `i` x y – make person x and person y friends. If they are already friends, no change needs to be made. If either x or y is a new person, add them.
- `d` x y – delete the friendship between person x and person y.
- `n` x – output the number of friends that person x has.
- `f` x – output the number of "friends of friends" that person x has. Notice that x and direct friends of x are not counted as "friends of friends."
- `s` x y – output the degree of separation between x and y. If there is no path from x to y, output `Not connected`.
- `q` – quit the program.

Sample Interaction

Input	Output	Explanation

`i`
`20`
`10`
	(no output)	Inserting a friendship causes no output.

`i`
`20`
`9`
	(no output)	Inserting a friendship causes no output.

`n` `20`	`2`	Person 20 has two friends (10 and 9)
`f` `20`	`3`	The friends of friends of 20 are 8, 11, 12.

`s`
`20`
`6`
	`4`	The shortest path is $20 \to 9 \to 8 \to 7 \to 6$.

`q`	(no output)	Program quits.