Canadian
Mathematics
Competition

An activity of The Centre for Education
in Mathematics and Computing,
University of Waterloo, Waterloo, Ontario

Canadian Computing Competition

for the Sun microsystems Awards

Tuesday, February 26, 2002

Problem J1

```
 _       _   _       _   _   _   _   _
| |   |  _|  _| |_| |_  |_    | |_| |_|
|_|   | |_   _|   |  _| |_|   | |_|  _|
```

Most digital devices show numbers using a seven segment display. The seven segments are arranged as shown:

```
   * * *
 *       *
 *       *
 *       *
   * * *
 *       *
 *       *
 *       *
   * * *
```

For this problem each segment is represented by three asterisks in a line as shown above. Any digit from 0 - 9 can be shown by illuminating the appropriate segments. For example the digit 1 may be displayed using the two segments on the right side:

```
         *
         *
         *

         *
         *
         *
```

The digit 4 needs four segments to display it properly:

```
 *       *
 *       *
 *       *
   * * *
         *
         *
         *
```

Write a program that will accept a single digit input from the user, and then display that digit using a seven segment display. You may assume that each segment is composed of three asterisks.

Sample Session. ***User input in italics***

```
Enter a digit between 0 and 9:
9
        * * *
      *       *
      *       *
      *       *
        * * *
              *
              *
              *
        * * *
```

Problem J2
AmeriCanadian

Americans spell differently from Canadians. Americans write "neighbor" and "color" while Canadians write "neighbour" and "colour”.

Write a program to help Americans translate to Canadian.

Your program should interact with the user in the following way. The user should type a word (not to exceed 64 letters) and if the word appears to use American spelling, the program should echo the Canadian spelling for the same word. If the word does not appear to use American spelling, it should be output without change. When the user types "quit!" the program should terminate.

The rules for detecting American spelling are quite naive: If the word has more than four letters and has a suffix consisting of a consonant followed by "or", you may assume it is an American spelling, and that the equivalent Canadian spelling replaces the "or" by "our". Note : you should treat the letter "y" as a vowel.

Keyboard input and screen output is expected.

Sample session. *User input in italics.*

```
Enter words to be translated:
color
colour
for
for
taylor
taylour
quit!
```

Problem S1J3
The Students' Council Breakfast

The students council in your school wants to organize a charity breakfast, and since older students are both wiser and richer, the members of the council decide that the price of each ticket will be based on how many years you have been in the school. A first year student will buy a PINK ticket, a second year student will buy a GREEN ticket, a third year student will buy a RED ticket, and a fourth year student will buy an ORANGE ticket.

Assume that all tickets are sold. Each colour of ticket is uniquely priced. Input the cost of a PINK, GREEN, RED, and ORANGE ticket (in that exact order). Input the exact amount of money to be raised by selling tickets. Output all combinations of tickets that produce exactly the desired amount to be raised. The combinations may appear in any order. Output the total number of combinations found. Output the smallest number of tickets to print to raise the desired amount so that printing cost is minimized.

Keyboard input and screen output is expected.

Sample Session. *User input in italics.*

```
Cost of PINK tickets:
1
Cost of GREEN tickets:
2
Cost of RED tickets:
3
Cost of ORANGE tickets:
4
How much must be raised with ticket sales?
3
Combinations are
# of PINK is 0    # of GREEN is 0     # of RED is 1    # of ORANGE is 0
# of PINK is 1    # of GREEN is 1     # of RED is 0    # of ORANGE is 0
# of PINK is 3    # of GREEN is 0     # of RED is 0    # of ORANGE is 0
Total combinations is 3.
Minimum number of tickets to print is 1.
```

Problem S2J4
Fraction Action

Many advanced calculators have a fraction feature that will simplify fractions for you.

You are to write a program that will accept for input a non-negative integer as a numerator and a positive integer as a denominator, and output the fraction in simplest form. That is, the fraction cannot be reduced any further, and the numerator will be less than the denominator. You can assume that all input numerators and denominators will produce valid fractions.

Keyboard input and screen output are expected.

Sample session. *User input in italics.*

```
Numerator
28
Denominator
7

4

Numerator
13
Denominator
5

2 3/5

Numerator
0
Denominator
7

0

Numerator
55
Denominator
10

5 1/2
```

S3J5
Blindfold

Rose and Colin are playing a game in their backyard. Since the backyard is rectangular we can think of it as a grid with *r* rows and *c* columns. Rose and Colin place obstacles on some of the squares.
The game is played as follows:
Colin covers his eyes with a blindfold then Rose carries him to some grid square in the backyard. She sets him down so that he is facing north, south, east or west. Colin does not know this initial position or direction.
Rose then instructs Colin to make a series of *m* moves through the backyard. Each move is one of: F - moves forward one grid square in the direction that he is facing, or
L - turns 90 degrees counter-clockwise, remaining on the same square, or
R - turns 90 degrees clockwise, remaining on the same square.

After making these moves, Colin is standing at some final position. He must now figure out where he is standing. You will help him by writing a program to determine all possible final positions. Assume that Colin's initial position, final position, and all intermediate positions lie within the backyard but never in a square that contains an obstacle. You may also assume that Colin is always facing a direction that is parallel to the sides of the backyard (north, south, east, or west).

The input begins with *r* and *c* ($1 \leq r \leq 375$; $1 \leq c \leq 80$), each on a separate line. Next are *r* lines of *c* characters describing the backyard: a period character denotes a grid square that Colin may walk through; an X character denotes a grid square with an obstacle. Below the grid is the number *m* ($0 \leq m \leq 30000$) followed by *m* lines describing Colin's moves. Each line has a single character: F, L, or R.

Your program should output the backyard grid, indicating all possible final positions with the * character.

Sample Input (Input file : blind.in)
```
2
4
....
.XX.
3
F
R
F
```

Output for Sample Input (Output file : blind.out)
```
.*..
.XX*
```

Problem S4
Bridge Crossing

A rope bridge traverses a deep gorge. People may cross the bridge in groups, but there is a limit (*M*) to the size of the group. The time taken for a group to cross is determined by the slowest member. You are responsible for safety on the bridge and part of your job is to control the groups crossing. People are waiting in a queue(line), when the previous group has crossed you tell the next few people they can now cross. Groups can be of different sizes; no group can contain more than *M* people, and the goal is to get everyone over in the shortest time, all the time maintaining the order of the people in the queue.

The first line of the input contains an integer M ($1 \le M \le 20$). The second line contains Q ($1 \le Q \le 100$), the length of the queue waiting to cross. For each person in the queue, a pair of input lines follows. The first of these is the name of the person, and the second is that person's individual time to cross the bridge. **Recall that a group crossing time will be the maximum crossing time for time for any individual in the group.**

Your output is to list the names of the people in each group, one group per line, which will obtain the minimum overall crossing time. If several groupings yield the same overall crossing time, any such grouping may be listed. The final line of the output is to give the total crossing time for the entire queue of people.

Sample input (Input file : bridge.in)

```
2
5
alice
1
bob
5
charlie
5
dobson
3
eric
3
```

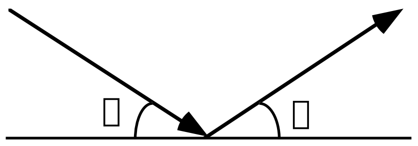
Output for Sample Input (Output file : bridge.out)

```
Total Time: 9
alice
bob charlie
dobson eric
```

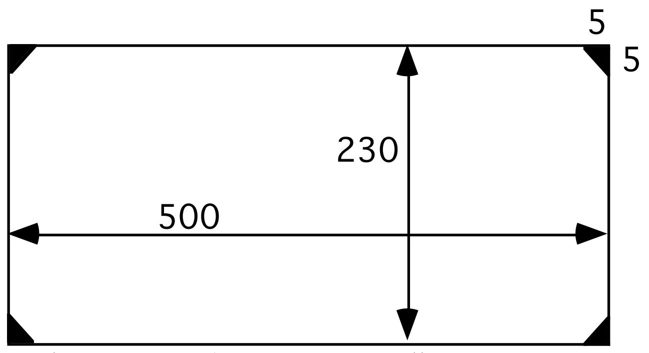
Problem S5
Follow the Bouncing Ball

Suppose you want to create a screen saver which has a ball bouncing around the edges. The action will continue until the ball hits a corner (like a pool table). At that point, the ball disappears off the screen and the screen goes black.

Balls bounce off the walls according to the laws of physics. The angle of the ball as it approaches the wall (the angle of incidence) will equal the angle of the ball as it bounces away (the angle of reflection).



Write a program which will determine how many bounces it will take before the bouncing ball will be swallowed by a corner, if at all. The screen dimensions are in the range [100, 1000] units. Consider the ball as a point on the screen. The corner pockets are 5 units along each wall, and if a ball hits the wall at the edge of a pocket it will continue bouncing.
A sample screen is shown below.



Input will consist of four integers, each on a separate line.

- n – the width of the screen, $100 \le n \le 1000$
- m – the height of the screen, $100 \le m \le 1000$
- p – the initial position of the ball at the bottom of the screen, $5 \le p \le n - 5$
- q – the position of the ball when it bounces off the right wall, $5 \le q \le m - 5$

The output consists of an integer which indicates the number of bounces the ball makes before it disappears in a corner. Count the hit off the right wall as the first bounce. It is possible that the ball will never bounce into a corner (assuming a frictionless surface), in this case, output should be 0. Note that when the ball hits a pocket is not counted as a bounce. You do not have to program the actual animation.

Sample Input (Input file : ball.in)

```
300
200
200
100
```

Output for Sample Input (Output file : ball.out)

```
2
```

Graeme,

This note was included in the contest package.

Note to Canadian Computing Competition Supervisors:

Please inform students writing the **Senior Division** of the CCC that in question **S5**,
p is the distance measured from the left side of the rectangle, and
q is the distance measured from the bottom of the rectangle.