



Canadian Mathematics Competition

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

Canadian Computing Competition

for the  Awards

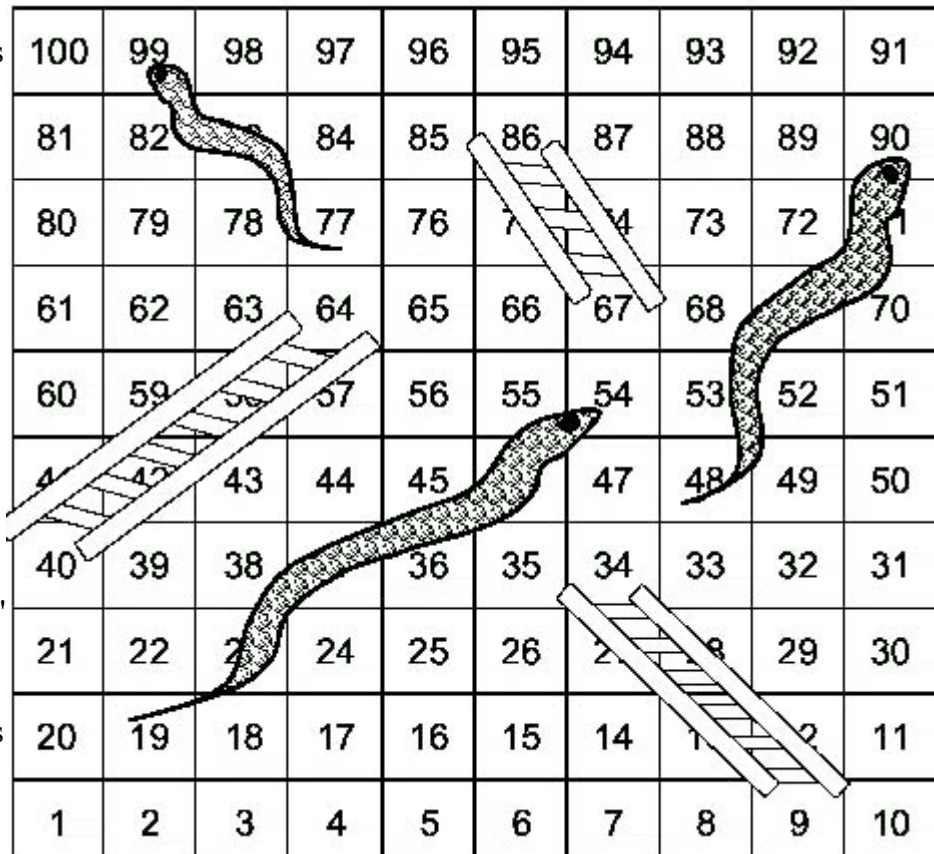
Tuesday, February 25, 2003

Sponsors:



Problem J3/S1 - Snakes and Ladders

Here (see illustration) is a game board for the game Snakes and Ladders. Each player throws a pair of dice to determine how many squares his/her game piece will advance. If the piece lands on the bottom of a ladder, the piece moves up to the square at the top of the ladder. If the piece lands on the top of a snake, the piece "slides" down to the square at the bottom of the snake. If the piece lands on the last square, the player wins. If the piece cannot advance the number of squares indicated by the dice, the piece is not moved at all.



100	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23	24	25	26	27	28	29	30
20	19	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

In order to help you play this game via a cell phone while travelling, you will write a program that simulates your moves on the board shown and, of course, runs on your handheld computer. You will repeatedly throw the dice and enter the result into the program. After each throw the program will report the number of the square where your piece lands.

When the program starts it should assume the piece is on square 1. It should repeatedly read input from the user (a number between 2 and 12) and report the number of the square where the piece lands. In addition, if the piece moves to the last square, the program should print "You Win!" and terminate. If the user enters 0 instead of a number between 2 and 12, the program should print "You Quit!" and terminate.

For clarity, you are to use the board pictured above and you should note that the board has 3 snakes (from 54 to 19, from 90 to 48 and from 99 to 77) and 3 ladders (from 9 to 34, from 40 to 64 and from 67 to 86).

Sample Session *User input in italics*

Enter sum of dice:

9

You are now on square 10

Enter sum of dice:

11

You are now on square 21

Enter sum of dice:

12

You are now on square 33

Enter sum of dice:

7

You are now on square 64

Enter sum of dice:

3

You are now on square 86

Enter sum of dice:

5

You are now on square 91

Enter sum of dice:

10

You are now on square 91

Enter sum of dice:

9

You are now on square 100

You Win!

Problem J4/S2 - Poetry

Input file: **poetry.in**

Output file: **poetry.out**

A simple poem consists of one or more four-line verses. Each line consists of one or more words consisting of upper or lower case letters, or a combination of both upper and lower case letters. Adjacent words on a line are separated by a single space.

We define the last syllable of a word to be the sequence of letters from the last vowel ("a", "e", "i", "o" or "u", but not "y") to the end of the word. If a word has no vowel, then the last syllable is the word itself. We say that two lines rhyme if their last syllables are the same, ignoring case.

You are to classify the form of rhyme in each verse. The form of rhyme can be *perfect*, *even*, *cross*, *shell*, or *free*:

perfect rhyme: the four lines in the verse all rhyme

even rhyme: the first two lines rhyme and the last two lines rhyme

cross rhyme: the first and third lines rhyme, as do the second and fourth

shell rhyme: the first and fourth lines rhyme, as do the second and third

free rhyme: any form that is not perfect, even, cross, or shell

The first line of the input file contains an integer N , the number of verses in the poem, $1 \leq N \leq 5$. The following $4N$ lines of the input file contain the lines of the poem. Each line contains at most 80 letters of the alphabet and spaces as described above.

The output should have N lines. For each verse of the poem there should a single line containing one of the words 'perfect', 'even', 'cross', 'shell' or 'free' describing the form of rhyme in that verse.

Sample Input 1

1

One plus one is small
one hundred plus one is not
you might be very tall
but summer is not

Output for Sample Input 1

cross

Sample Input 2

2

I say to you boo

You say boohoo

I cry too

It is too much foo

Your teacher has to mark

and mark and mark and teach

To do well on this contest you have to reach

for everything with a lark

Output for Sample Input 2

perfect

shell

Sample Input 3

2

It seems though

that without some dough

creating such a bash

is a weighty in terms of cash

But how I see

the problem so fair

is to write subtle verse

with hardly a rhyme

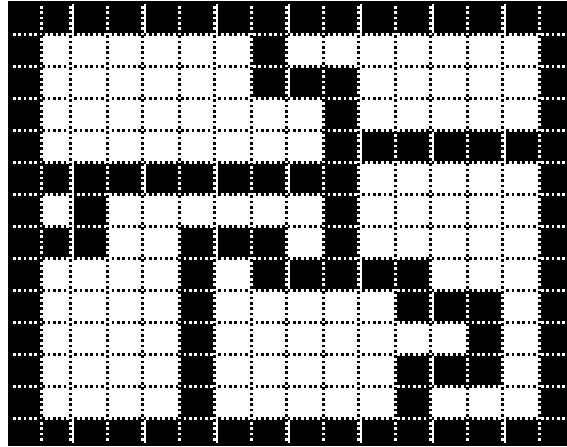
Output for Sample Input 3

even

free

Problem J5/S3 - Floor Plan

The floor plan of a house shows rooms separated by walls. This floor plan can be transferred to a grid using the character “I” for walls and “.” for room space. Doorways are not shown. Each “I” or “.” character occupies one square metre.



In this diagram there are six rooms.

You have been given the floor plan of a house and a supply of hardwood flooring. You are to determine how many rooms will have the flooring installed if you start installing the floor in the largest room first and move to the next largest room, and so on. You may not skip over any room, and you must stop when you do not have enough wood for the next room. Output the number of rooms that can have hardwood installed, and how many square metres of flooring are left over. No room will be larger than 64 square metres.

The first line of the data file contains the number of square metres of flooring you have. The second line in the file contains an integer r from 1 – 25 that represents the number of rows in the grid. The third line contains an integer c from 1 – 25 that represents the number of columns in the grid. The remaining r lines contain c characters of grid data.

Sample Input 1

```
105
14
16
IIIIIIIIIIIIIIIIII
I.....I.....I
I.....III.....I
I.....I.....I
I.....IIIIIII
IIIIIIIIII.....I
I.I.....I.....I
III..III.I.....I
I....I.IIIII...I
I....I.....III.I
I....I.....I.I
I....I.....III.I
I....I.....I...I
IIIIIIIIIIIIIIIIII
```

Output for Sample Input 1

4 rooms, 1 square metre(s) left over

Sample Input 2

```
13
2
3
.I.
.I.
```

Output for Sample Input 2

2 rooms, 9 square metre(s) left over

Problem S4 - Substrings

Input file: **substr.in**

Output file: **substr.out**

How many distinct substrings does a given string S have?

For example, if $S = \text{"abc"}$, S has 7 distinct substrings: $\{\text{"", "a", "b", "c", "ab", "bc", "abc"}\}$. Note that the empty string and S itself are considered substrings of S .

On the other hand, if $S = \text{"aaa"}$, S has only 4 distinct substrings: $\{\text{"", "a", "aa", "aaa"}\}$.

The first line of the input file contains N , the number of test cases. For each test case, a line follows giving S , a string of from 1 to 1000 alphanumeric characters. Your output consists of one line per case, giving the number of distinct substrings of S . Try to write an efficient program.

Sample Input

```
2
abc
aaa
```

Output for Sample Input

```
7
4
```


Problem S5 - Trucking Troubles

Input file: **truck.in**

Output file: **truck.out**

You are a salesperson selling trucks which can carry trucks which can carry trucks. Needless to say, your trucks are heavy. Also needless to say, you have to drive one of these trucks across a wide wet domain, and since it is wet, you need to drive over some bridges. In fact, on every road between two cities, there is a bridge but there is not a direct road between every pair of cities. Each bridge can support a certain maximum weight. This maximum weight is an integer from 0 to 100000.

You have been given a list of cities where there are customers who are eager to view one of your trucks. These cities are called *destination cities*. Since you must decide which truck you will drive through these cities, you will have to answer the following problem: what is the maximum weight that can be driven through these destination cities? You are to write a program to solve this problem.

The first line of input will contain three positive integers: c , r and d specifying the number of cities (in total), number of roads between cities and number of destination cities, respectively. The cities are numbered from 1 to c . There are at most 1000 cities and at most 10000 roads. The next r lines contain triples $x\ y\ w$ indicating that this road runs between city x and city y and it has a maximum weight capacity of w . The next d lines give the destination cities you must visit with your truck. There will be at least one destination city.

You can assume that you are starting in city 1 and that city 1 is not a destination city. You can visit the d destination cities in any order, but you must visit all d destination cities.

The output from your program is a single integer, the largest weight that can be driven through all d destination cities.

Sample Input

```
5 7 3
1 2 20
1 3 50
1 4 70
1 5 90
2 3 30
3 4 40
4 5 60
2
4
5
```

Output for Sample Input

```
30
```