UTD High School Programming Contest 11-8-2014
Advanced Category

- **Time Allowed: three hours.**
- **Each team must only use one of UTD's computers.**
- **Answer the questions in any order.**
- **Use only Java 1.7, minGW g++, or MS Visual Studio C/C++**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from a named file and output to System.out (cout for C/C++ programmers.)**
- **Do not access the web.**
- **Do not use any recording device containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client.**

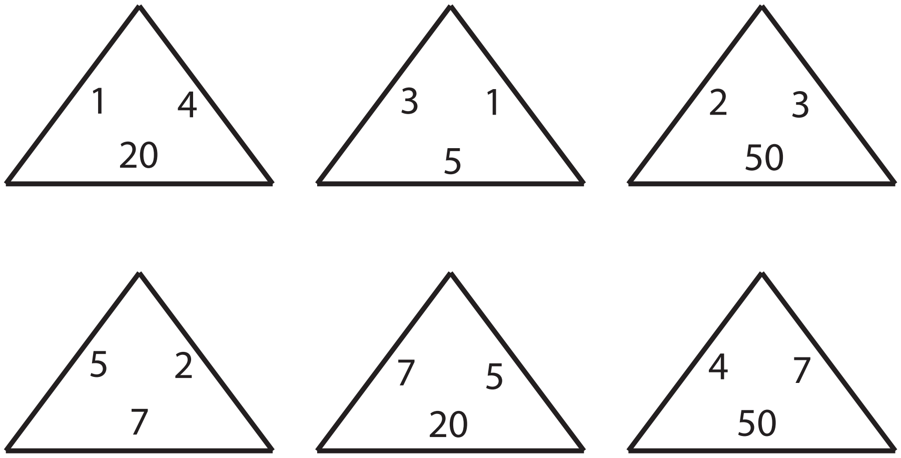
Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

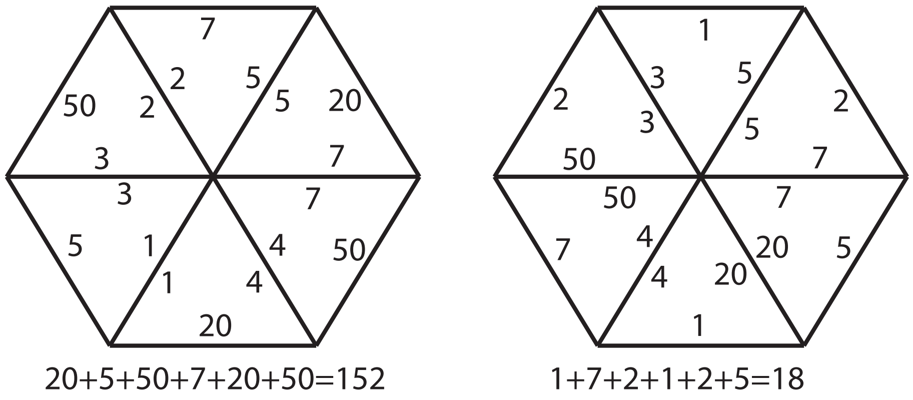
In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

A. **Triangles**

Input File: A.txt
Run Time Limit: 10sec



In the triangle game you start off with six triangles numbered on each edge, as in the example above. You can slide and rotate the triangles so they form a hexagon, but the hexagon is only legal if edges common to two triangles have the same number on them. You may not flip any triangle over. Two legal hexagons formed from the six triangles are illustrated below.



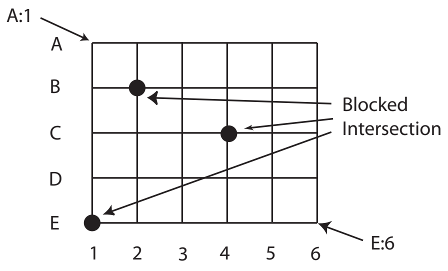
The score for a legal hexagon is the sum of the numbers on the six outside edges.

Your problem is to find the highest score that can be achieved with any six particular triangles. The input file will contain up to ten datasets. Each dataset is a sequence of six lines. On each line will be three integers in the range [1,100] separated by space characters. These three integers are the numbers on the triangles given in clockwise order. Datasets are separated by a line containing only an asterisk. The last dataset is followed by a line containing only a dollar sign. For each input data set, the output is a line containing only the word “none” if there are no legal hexagons, or the highest score if there is a legal hexagon.

Example Input	Output for Sample Input
1 4 20	152
3 1 5	21
50 2 3	none
5 2 7	
7 5 20	
4 7 50	
*	
10 1 20	
20 2 30	
30 3 40	
40 4 50	
50 5 60	
60 6 10	
*	
10 1 20	
20 2 30	
30 3 40	
40 4 50	
50 5 60	
10 6 60	
$	

B. **Hatmando**

Input File: B.txt
Run Time Limit: 10sec



The town of Hatmando has a street layout that follows the Manhattan style. East-west streets are equally spaced and parallel. Their names are upper-case alphabetical characters. North-south streets are also equally spaced and parallel. Their names are integers beginning with 1. Above is an example town map illustrating the street and intersection naming conventions.

There are no one-way streets. Occasionally an intersection is completely closed to traffic so that repairs can be made to the marmalade supply pipes.

Citizens of Hatmando complain bitterly to the town mayor about these road repairs, so he asks you to write a program to calculate how many shortest routes exist from the North-West corner of town to the South-East corner of town. He plans to publish the results in the town newspaper to show how open the town is to traffic. Given a town map with a list of closed intersections, find the number of shortest paths between the two given points. The mayor asks that you only consider routes made up of south-going and east-going segments.

Input:
Input begins with a line containing N, the number of datasets to follow ($1 \leq N \leq 20$). Each data set has the following lines:

- Two space-separated integers giving the number of east-west streets and the number of north-south streets, both in the range 1 to 20
- A line beginning with an integer M ($0 \leq M \leq 20$) giving the number of blocked intersections, followed by a space and then M space-separated intersection designations.

Output:
For each dataset, output a single line giving the number of shortest routes from the North-West corner of town to the South-East corner of town.

Sample Input	**Output for Sample Input**
3 4 4 2 C:3 A:2 8 8 5 C:3 A:2 D:4 D:5 E:1 8 8 0	4 490 3432

C. **Digits**

Input File: C.txt
Run Time Limit: 10sec

An hour on the TRE from Dallas to Fort Worth was sufficient time to develop this silly game.

Start with a positive integer S. So long as it has more than one digit, compute the product of its digits and repeat. For example, if starting with 95, compute $9 \times 5 = 45$. Since 45 has more than one digit, compute $4 \times 5 = 20$. Continuing with 20, compute $2 \times 0 = 0$. Having reached 0, which is a single-digit number, the game is over.

As a second example, if we begin with 396, we get the following computations:
$3 \times 9 \times 6 = 162$
$1 \times 6 \times 2 = 12$
$1 \times 2 = 2$
and we stop the game having reached 2.

Input:
The input contains up to 50 lines. On each line is a single integer $1 \leq S \leq 100000$, designating the starting value. The value S will not have any leading zeros. A value of 0 designates the end of the input.

Output:
For each nonzero input value, a single line of output should express the ordered sequence of values that are considered during the game, starting with the original value.

Sample Input	**Output for Sample Input**
95	95 45 20 0
396	396 162 12 2
28	28 16 6
4	4
40	40 0
0	

D. **Juggler Sequence**

Input File: D.txt
Run Time Limit: 10sec

A juggler sequence has nothing to do with juggling. Given a number a_k the next number in the sequence is defined by the recurrence relation:

$$a_{k+1} = \begin{cases} \lfloor a_k^{\frac{3}{2}} \rfloor & : \quad a_k \textit{ is odd} \\ \lfloor a_k^{\frac{1}{2}} \rfloor & : \quad a_k \textit{ is even} \end{cases}$$

For example, if $a_0 = 3$ then the Juggler Sequence is 3, 5, 11, 36, 6, 2, 1. If $a_k = 1$ then all subsequent terms in the sequence are 1.

It is conjectured that, for all starting values, the sequence eventually reduces to 1, but a proof of this has not been found.

Given a starting value, a_0, print the number of terms in the corresponding Juggler Sequence and print the highest value obtained. Values in these sequences can be quite large. For example, for $a_0 = 48,443$ the largest value in the Juggler Sequence has 972,463 decimal digits. Fortunately you will not have to worry about such large numbers. Each term in a juggler sequence in this problem will not exceed 15 decimal digits.

Input:
The input will contain several test cases. The first line of input contains the value N $(1 \leq N \leq 20)$ giving the number of test cases. Then N lines follow, each one containing a value for a_0.

Output:
For each starting value, a_0, print a line containing two integers: the length of the Juggler Sequence (including a_0 and the first value of 1 in the sequence) followed by a space, and then the largest value found in the sequence.

Sample Input	**Output for Sample Input**
4	2 2
2	7 36
3	3 6
6	8 36
10	

E. **Roman Calculator**

Input File: E.txt
Run Time Limit: 10sec

A Roman numeral consists of a set of letters of the alphabet. Each letter has a particular value, as shown in the following table:

Letter	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Generally, Roman numerals are written in descending order of value from left to right and are added sequentially. However, certain pairs of symbols employ a subtractive principle. If a symbol of smaller value precedes a symbol of larger value, the smaller value is subtracted from the larger value, and the result is added to the total. This subtractive principle has the following rules:

"I" may only precede "V" and "X" (e.g., IV=4).
"X" may only precede "L" and "C" (e.g., XC=900).
"C" may only precede "D" and "M".
"V", "L" and "D" are never followed by a symbol of higher value.
A subtractive sequence may not include more that 2 symbols. For example, "IXL" is not legal since it could represent 50-(10-1)=41 or 50-10-1=39.

Symbols "I", "X", "C" and "M" cannot appear more than three consecutive times. Symbols "V", "L" and "D" cannot appear more than once consecutively.

Roman numerals cannot represent the number zero. For values greater or equal than 4000 the Romans used bars placed above the letters to indicate multiplication by 1000. In this problem no input, intermediate, or final result will be greater than 4000.

Write a program that reads arithmetic expressions of the form

Roman Numeral $\square$ Roman Numeral

where $\square \in \{+, -, *, /\}$ and prints the answers in Roman Numerals. Use only upper case letters for the Roman numerals.

Input:

The input consists of up to 20 lines, each one containing an arithmetic expression of form $x\square y$, where $\square \in \{+, -, *, /\}$ and x, y are integers expressed in Roman Numerals, each with decimal value in $[1, 3999]$.

All result values will be in $[1, 3999]$. Division must round toward zero.

Output:

For each input line print one line of output containing the result of the arithmetic expression in Roman Numerals.

Sample Input	**Output for Sample Input**
MCMLXXXIII-XXV	MCMLVIII
IV+CMXLII	CMXLVI

F. **FourD**

Input File: F.txt
Run Time Limit: 10sec

It's the day after the fall holidays in the land of FourD. It's time to nest the gift boxes to get them ready for the trash collector. Although hard for us to visualize, in this land every object has four physical dimensions. For example, a box may have dimensions (2,3,4,5). Even more strange is that objects may easily be morphed so that their dimensions are in any required order. Just give our (2,3,4,5) box a squeeze and it becomes a box with dimensions (3,5,2,4), or (5,4,2,3).

A box D with dimensions (d_1, d_2, d_3, d_4) fits inside a box G with dimensions (g_1, g_2, g_3, g_4) if there is a permutation of the dimensions of D such that in each dimension, the value in D is less than the corresponding value in G. We write $D < G$.

For example, $(2, 3, 4, 5)$ fits into $(3, 4, 5, 6)$ and into $(5, 8, 5, 3)$ (since $(2, 3, 4, 5)$ can be permuted to form $(4, 5, 3, 2)$ and $(4, 5, 3, 2) < (5, 8, 5, 3)$).

Given a list of 4D box dimensions, find the longest nesting sequence.

Input
The input consists of a series of up to 10 datasets. Each dataset begins with a line containing the the number of boxes k.

This line is followed by k lines, one line per box, each containing 4 space-separated integer dimensions. All measurements will be in the range[1,999]. The maximum number of boxes in a dataset is 20.

Input is terminated by End Of File.

Output
For each box sequence in the input file, output the length of the longest nesting string followed on the next line by a list of the boxes that comprise this string as follows: The "smallest" or "innermost" box of the nesting string should be listed first, the next box (if there is one) should be listed second, etc. The boxes should be numbered according to the order in which they appeared in the input file (the first box is box 1, etc.). If there is more than one longest nesting string then output the numerically smallest one. For example, 1 2 8 5 is smaller than 1 2 9 4 (1285<1294).

Sample Input	Output for Sample Input

4
2 3 4 5
1 4 2 6
8 8 8 8
7 6 3 2
5
6 6 7 8
1 2 3 4
9 10 11 12
2 2 3 3
3 3 3 3
	3
	2 4 3
	3
	2 1 3

G. **Quick Search**

Input File: G.txt
Run Time Limit: 10sec

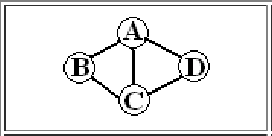
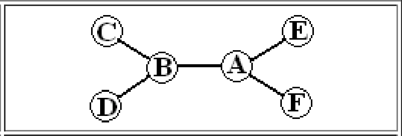
	

Figure 1	Figure 2

A police unit has dealt with a number of situations where they need to search multiple sites as rapidly as possible with a small group of officers. Figures 1 and 2 are diagrams for two situations. Sites to search are labeled with capital letters. The site labeled A is the common starting point. Some of these locations have connecting paths. Assume each connecting path takes one unit of time to traverse, and individual sites take no time for a visual search.

Consider Figure 1. If there are 3 or more officers it takes only one unit of time to search. Different officers follow paths AB, AC, and AD. If there are two officers, the complete search will take two time units. For example one officer could follow the path ABC and the other AD.

Consider Figure 2. If there were 3 officers, they could follow paths ABC, ABD, and AEAF and take 3 units of time. With 2 officers they could follow paths ABCBD and AEAF, and take 4 units of time.

These are simple enough examples to figure out by hand. For more complicated arrangements they want your programming help.

Input:
The input will consist of 1 to 25 datasets, followed by a line containing only 0.

A dataset is a single line starting with space-separated positive integers $s \quad n \quad p$, where s is the number of sites to search, n is the number of officers, and p is the number of connecting paths between search sites. Limits are $2 \leq s \leq 10$, $1 \leq n \leq 4$, and $p \leq 20$. The rest of the line contains p pairs of capital letters, indicating paths between sites, each preceded by a space character.

Sites are labeled with the first s capital letters. All sites will be connected by some sequence of connecting paths. The two letters in each letter pair will be in increasing alphabetical order. No letter pair will be repeated.

Output:
There is one line of output for each dataset, containing only the minimum search time for n officers starting at site A.

The sample input data sets correspond to the scenarios discussed.

Sample Input
4 4 5 AB BC CD AD AC
4 2 5 AB BC CD AD AC
6 3 5 AB BC BD AE AF
6 2 5 AB BC BD AE AF
0

Output for Sample Input
1
2
3
4