Nada Open
2004-02-14

Problems

Requirements and friendly advice

Do not write to standard error

Remember "`return 0`" in C and C++

C++: `iostream` is much slower than `cstdio`

Java: The filename must be the same as the name of the main class

Maybe we should have killed some of our "darlings," but we didn't have the heart. Not on Valentine's day.

Problem A: Pulse function median

Problemsetter: Andreas Björklund

Problem

A function f is a sum of, possibly overlapping, triangular pulses. You are to find the median z. More precisely, if f is a function and

$$A = \int_{-\infty}^{\infty} f(x)\,dx$$

exists and is finite, then z is the median if

$$A/2 = F(z) \stackrel{\text{def}}{=} \int_{-\infty}^{z} f(x)\,dx$$

If there are several possible values for z (can happen if $f(x) = 0$ on some interval) then any z satisfying the equation qualifies as the median.

A pulse is described by three floating point numbers (s, w, h) where w and h are strictly greater than zero. The pulse is the function

$$g(x) = \begin{cases} 0 & \text{if } x < s \text{ or } x > s + w \\ 2h(x - s)/w & \text{if } s \le x < s + w/2 \\ 2h(1 - (x - s)/w) & \text{if } s + w/2 \le x \le s + w \end{cases}$$

That is, the pulse is triangular with height h and width w, and it starts at $x = s$.

Finally, if we have m pulses $g_1, \ldots, g_m$, then the function f is

$$f(x) = \sum_{1}^{m} g_i(x)$$

Input

The first line of the input contains an integer N which is the number of test cases. After that come N blocks of input. A block consists of an integer M on a separate line, giving the number of pulses ($1 \le M \le 5000$). Then follow M lines, each containing three numbers S W H describing a pulse and the function is the sum of these pulses (see above). You may assume that $S \ge 0$ and that both W and H are strictly greater than 0.

Output

For each test case output the median (as defined above): a single floating point number.

(Note: To avoid problems due to rounding, we will check that your z satisfies $|2F(z) - A|/A < \epsilon$ for some small ϵ.)

Sample input

```
2
1
0.0 5.0 1.3
2
0.0 5.0 1.0
1.5 3.0 2.0
```

Sample output

```
2.5
2.834733547569204
```

Problem B: Broken games

Problemsetter: Stefan Nilsson

GamesForYou wants to create a line of unique games, each and every single copy should be different.

A game consists of a rectangle of size $m \cdot n$, where both m and n are positive integers. The rectangle is divided into mn squares. A square may be empty or it may contain a movable tile marked with a letter. The object of the game is to move the tiles around until a final configuration is achieved. It's only possible to move a tile one step at a time and only to an empty square situated either directly to the left, right, above or below the current position.

The R&D department of GamesForYou soon realized that some games couldn't be solved and they decided to hire a computer scientist to write a program that could distinguish the solvable games from the unsolvable ones.

The classic version of this game is the famous 15-puzzle by Sam LLoyd (1841-1911) using a 4 by 4 square with pieces numbered 1 to 15 and a single empty square. He offered $1000 to anyone who could get all the numbers in order starting from a configuration where only the pieces 14 and 15 had been switched. Nobody claimed the prize sine there is no solution for this case.

Problem

You will be given as input two configurations. The program should decide if it's possible to go from one configuration to the other by moving the tiles. A configuration is described by a rectangle of letters ('A' - 'Z') and empty positions ('.'). Note that the problem will have several test cases.

Input

The first line gives you the number of test cases. This is followed by a blank line. Each test case consists of two rectangles of characters ('A' - 'Z' and '.'), one rectangle describes the initial configuration and the other one describes the final configuration. Every rectangle is followed by a blank line. You can safely assume that all lines describing a single rectangle have the same length and that no rectangle has zero rows or zero columns. Also, no rectangle has more than 35 rows or more than 35 columns.

Output

For each test case output either "YES" or "NO" depending on whether the game is solvable or not.

Sample input

```
2

ZY
X.

XY
Z.

AABC
DEFG
```

```
HI..

..IH
GFED
CBAA
```

Sample output

```
NO
YES
```

Problem C: Bilateral projects

Problemsetter: Mikael Goldmann

A friend of yours works at a fairly large company that has offices in Stockholm and London. Cooperation between the offices in the two countries is extensive and the situation is that each of the many but small projects are handled by a two-person team with a member in each city. While emails, faxes, and phones are wonderful, and work well within each team, the CEO wants a briefing every year on every project. For this purpose the CEO invites representatives from the project to Barbados for a week during which he gets presentations of all the projects as well as a chance to relax in the sun.

Problem

Money is tight and a new policy has been created. The CEO wants to invite as few people as possible. You are to write a program that, given all the two-person teams, computes the smallest number of pepople that must be invited in order to get at least one person from each project. It should also output a possible set of people to invite.

Input

The first line contains an integer which is the number of test cases. Each test case starts a line with an integer $m \geq 1$ which is the number of two-person teams. The following m lines each contain two integers, i, j separated by a space, being the employee IDs of the two employees in that team (the first one is from Stockholm and the second one is from London). Stocholm employees have IDs in the range 1000 to 1999 and London employees have IDs in the range 2000 to 2999. An employee can be a member of several teams, but there cannot be several teams consisting of the same pair of employees. There are at most 200 teams in a single test case.

Output

For each test case output a first single line with an integer k indicating the smallest number of employees that must be invited to meet the requirements above. On following line, output the IDs of k employees (separated by spaces) such that for each two-person team, at least one of the IDs in that team appears on this line (if there are several optimal solutions it does not matter which one of them you output).

Sample input

```
2
4
1000 2003
1100 2002
1100 2003
1001 2002
6
1000 2000
1001 2000
1002 2000
1002 2001
```

```
1002 2002
```

Sample output

```
2
2002 2003
2
2000 1002
```

Problem D: Tournament scheduling

Problemsetter: Stefan Nilsson

Your task, should you decide to accept it, is to create schedules for tournaments.

Problem

The following rules apply. A game is played between two teams. Any number of games may be scheduled on a single day; however, no team can play more than one game per day. Every team should play against every other team once. The number of days should be as small as possible.

Input

The first line gives you the number of tournaments. The number of teams n, where $2 \leq n \leq 20$, in each tournament is written on a separate line.

Output

For each tournament output a schedule in the following format. The teams are assigned consecutive letters from the alphabet starting at 'a'. A game between 'x' and 'y' is written 'xy'. Games played on the same day should be separated by commas. Games played on separate days should be separated by semicolons. Tournaments should be separated by line breaks.

Sample input

```
2
3
4
```

Sample output

```
ab;ac;bc
ab,cd;ac,bd;ad,bc
```

Problem E: Verifying output

Problemsetter: Mikael Lagerkvist

Occam is a programming language designed to be used in conjunction with the transputer processor. The language is highly parallell, and uses mesage passing as the communication primitive. The messages that are sent are sent using asynchronous communication, with (for our purposes) unbounded buffers.

One interesting characteristic of occam is that programs aren't deterministic. This means that a program written in occam may behave differently during different runs of the program. To examine this, you will write a program that, for some simple occam programs (i.e. programs without variables, channels, loops, functions and so on), determines if the supplied output is a possible output of the program.

occam/no

The following is a description of a small simplified subset of the language, called occam/no. The language consists of parallel and sequential composition, send's and receive's to and from channels, alternation between receives and printing. The description uses BNF-notation, where

```
processlist     ::= process ';'
                    processlist
                |   process
```

should be read as "a processlist is either composed of a process, followed by a semicolon and then another processlist, or it is composed of a process".

```
program         ::= process
```

An occam/no program consists of a single process. The program ends when the process ends.

```
process         ::= output
                |   sequence
                |   parallell
```

A process is either an `output`-statement or a `sequence`- or `parallell` constructed process.

```
sequence        ::= 'SEQ'
                       processlist
                    'END'
```

A `sequence` is a composition of processes that will be executed in turn, from the first to the last (this equals the normal composition of statements in C or Java using semicolons), where each process starts executing after the previous has ended. The `sequence` ends when the last process ends.

```
parallell       ::= 'PAR'
                      processlist
                    'END'
```

The processes in a `parallell`-block is executed in pseudo-parallell. That means that one of the processes will take one step, and then another process takes one step, and so on. Which of the processes that gets to execute in each turn is undefined. The `parallell` ends when all processes has ended.

One step is defined as a single `output` statement.

```
processlist     ::= process ws ';' ws
                    processlist
                |   process
```

A `processlist` is a list of processes separated by semicolons.

```
output          ::= 'write_line' '(' ws literalnum ws ')'
```

An `output` statement means that the value of the `literalnum` will be written on a line by itself to the standard output.

```
literalnum      ::= a literal integer number, e.g. 124770 or 14986124
```

A `literalnum` is a literal integer number, in the form $[0-9]*$, with a value in the range $[0, 2^{31})$

```
ws              ::= at least one whitespacecharacter, possibly more.
```

A `ws` is some whitespace (i.e. space, tab or newline), i.e. on the the form [
t
n]+.

Problem

Given an occam/no program and an output, determine wheter the output could have been generated by the program. You are guaranteed that the program will not output one integer more than once.

Input

The first line contains an integer $N > 0$ which is the number of test cases.

The following lines contains N pairs of program and output that are to be checked against each other. The testcases are separated by newlines. First in each testcase comes an integer l_1 which describes the number of lines of the program. After that, the next l_1 lines contains the program. Then comes an integer l_2 describing the number of lines in the proposed output from the program. After that, the next l_2 lines contains the proposed output.

Output

Output either “yes” or “no” on a line by itself, indicating wheter the proposed output could have been generated by the program or not.

Sample input

```
2
4
PAR
  write_line( 18 ) ;
  write_line( 2 )
END
2
2
18

1
write_line(1);
1
2
```

Sample output

```
yes
no
```

Problem F: Counter action

Problemsetter: Mikael Goldmann

A new hardware company, Untel, are investigating the ramifications of using registers holding integers in bases that are not two. They experiment with ternary logic using terts (ternary digits) or quarts (base 4) and so on, instead of bits. Initially they are concerned with counters, i.e., registers holding an integer and for which there is an operation `incr` that increments the contents by 1. It turns ot that the cost of an `incr`-operation depends on the contents of the counter and the base used. The cost is equal to the number of digits in the register that are changed by the operation. For example, if a base-3 register contains 201 (written in base 3), then `incr` will cost 1 since only the last tert changed, but if it contins 222, the cost is 4, since the result is 1000 after `incr` (the three 2s and an implicit leading 0 have all changed this time).

Problem

Untel want you to calculate the cost of performing a series of `incr`-instructions on a base-c register (for various choices of c and various initial values in the register).

Input

The input consists of an integer on a line by itself giving the number of test cases. Each test case consists of three integers a, b, and c (in base 10) on the same line, separated by at least one space. You can assume that $0 \leq a, b$, that $a + b \leq 10^9$, and that $2 \leq c \leq 32$.

Output

For each test case a, b, c ouput the cost of incrementing b times a base c counter that initially contains a.

Sample input

```
2
1 2 2
0 100 10
```

Sample output

```
3
111
```

Problem G: 12-Puzzle

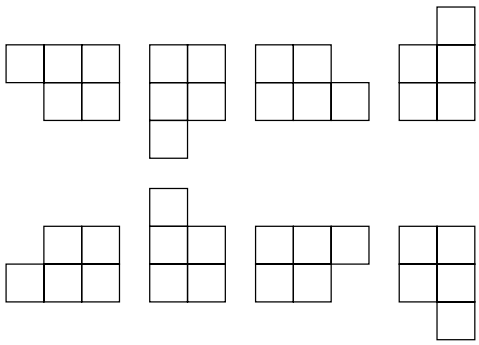
Problemsetter: Mikael Goldmann

The other day my kids found a puzzle that had been tucked away in a drawer for years. It seems easy enough as it has only twelve pieces, but an hour later they still had not figured it out and until it has been solved it does not fit in its box. Unfortunately it has been years since I solved it and I do not remember how, so it is still lying about. Please help me solve it so I can tidy up.

Problem

Your task is to find a legal solution to the puzzle. The puzzle box is rectangular with the sides 5 and 6, so it holds 30 "squares". Each of the twelve pieces has an area that is five squares big. The twelve pieces should be arranged into two 5 by 6 rectangles that fit on top of each other in the box.

Each piece is unique, but there is no picture so they can be rotated 0, 90, 180 or 270 degrees. They can also be flipped over. The picture shows an example of a piece and all its possible orientations.



I have named the pieces A through L, and below they are represented as patterns of the corresponding letters (the piece in the picture is piece F):

```
 A  BB   CCCCC DDDD EEEE FFF GG  HHH I   JJJ  KKK LL
AAA  BB        D     E   FF   GG H   III   JJ K K  L
 A   B                       G H   I             LL
```

There are many ways to select six pieces and build a rectangle, but for almost none of them is it possible to build a second rectangle with the remaining six. Still, that is the problem I need help with!

Input

There is no input for this problem. However, the patterns for the pieces are in the file `12puzzle.in` on your account.

Output

The solution is a *text file* formatted as follows. There are 11 lines with 6 uppercase letters on each line. The top 5 lines represent a rectangle made out of 6 pieces. The 6th line is blank, and the last 5 lines represent the second rectangle. Each puzzle piece is represented by 5 letters (piece A by As, and so on), indicating which piece occupies a sub square of a rectangle (each piece obviously occupying 5 sub squares).

Both rectangles have 5 rows and 6 columns. Each piece should be used exactly once. Pieces may be rotated and/or reflected as indicated in the picture, but they may not be altered in other ways. Below is an example of the file format, but it is not a legal solution since it does not use every piece exactly once.

```
KKKHHH
KEKLLH
EEEELH
FFJJLL
FFFJJJ

IIIBBJ
DIBBJJ
DIABJH
DAAAJH
DDAHHH
```

There are several solutions, and any legal solution will do.

Problem H: The power of substitution

Problemsetter: Mikael Goldmann

A substitution cipher uses a substitution table that for each letter x in the alphabet assigns another (possibly the same) letter $p[x]$ in the same alphabet. A message $m_1\, m_2\, m_3\, m_4$ is then encrypted as $E_p(m_1\, m_2\, m_3\, m_4) = p[m_1]\, p[m_2]\, p[m_3]\, p[m_4]$. This only works if p is one-to-one. That is, p has the property that $p[x] = p[x']$ only if $x = x'$.

Now consider the idea of iterating this, i.e., to encrypt the encryption by applying the substitution table again. We can therefore define substitution to the power k, denoted $E_p^k()$ by

$$\begin{aligned} E_p^0(m_1 \ldots m_\ell) &= m_1 \ldots m_\ell \\ E_p^{k+1}(m_1 \ldots m_\ell) &= E_p(E_p^k(m_1 \ldots m_\ell)) \end{aligned}$$

Now, given $m_1 \ldots m_\ell$, $c_1 \ldots c_\ell$, and $p[]$, we might ask (and we will!) for a k such that $E_p^k(m_1 \ldots m_\ell) = c_1 \ldots c_\ell$.

Problem

In our case the alphabet will have 100 symbols denoted 1, 2,..., 100. You will be given a message, m, a cryptotext, c, and a substitution table, p. You are to find a k such that $E_p^k(c) = d$. Specificly, we want such a k in the interval $0 \le k \le 10^9$ (there will always be a solution for the test cases).

Input

The input starts with an integer N on a line by itself. This is the number of test cases.

A test case consists of four lines: first a single integer L which is the size of the message (at least 1, at most 200), second a line with L numbers (the message m), third a line with L numbers (the crypto text c) and finally a line with 100 numbers giving the substitution table $p[]$ (the first number is $p[1]$, the second is $p[2]$ and so on).

You can assume that the numbers in the message and crypto text are all integers that are at least 1 and at most 100. You can assume the same for p, and furthermore that p is one-to-one.

Output

For each test case output a single integer k on a line by itself. This k should be non-neagative and at most 10^9, and obviously satisfy $E_p^k(m) = c$.

Sample input

```
2
4
1 2 3 4
11 12 13 14
2 3 4 5 6 7 ..... 99 100 1
3
1 2 10
2 1 9
2 1 4 3 6 5 8 7 ..... 98 97 100 99
```

Note that some elements in p have been left out for readbility. However, the pattern should be clear. In the test input there are four lines for every test case.

Sample output

```
10
1
```

Problem I: The greedy vizier

Problemsetter: Mikael Lagerkvist

> Congratulations!
>
> You have been selected to marry one of the Sultan's *n* daughters! Please report to the palace tomorrow at noon.
> You will be presented to one daughter at a time, and will be told her dowry, after which you may decide either to propose to her or not. If choose not to propose, another daugther will be presented, and so on. You may not go back in the line of daughters, and you must ask for one daughter's hand in marriage!
>
> Iznogoud
> Vizier

When you read this letter, you immediately get suspicious. Iznogoud is never up to any good, and this sounds to good to be true. After all, who doesn't want to marry one of the Sultan's daughters? After some snooping around, you realize that the vizier is very greedy, and that he will prevent your future marriage with the daughter of your choice, unless you choose the one with the lowest dowry. Since his method of prevention is cutting of your head, you must do your utmost to increase your chances of selecting the daughter with the lowest dowry.

After some extensive meditation, you realise that you must learn something about the dowrys before you propose to a daughter. This, you do by declining the first k daughter's (for some suitable choice of k), and then taking the first one with a lower dowry than any you've seen so far.

Problem

Given a list of the n daughters names and dowrys in the order that they are presented to (un)lucky suitor, output which daughter the suitor would select, if he wants to maximize his chance of survival based upon his limited knowledge (the number of daughters and the dowrys of the daughters that has been presented). The order the daughters are presented in is random.

Input

The first line contains an integer $N > 0$ which is the number of test cases.

Each of the following N test cases begins with a single integer $2 < n < 300$ specifying the number of daughters. After that, there are n lines composed of a, for this testcase unique, name containing no whitespace followed by a space and then the value of the dowry in Sultanese Dihnars.

Output

For each testcase, output the name of the daughter the smart suitor would choose.

Sample input

```
1
5
Lee 200
Vel 400
Fleu 600
Hel 800
Lin 1000
```

Sample output

```
Lin
```