

Examen

0 Quelques directives

- Vous n'avez droit qu'à un seul ordinateur par groupe.
- Vous n'avez droit qu'aux pages sur la STL ou les librairies standards, vous ne pouvez pas aller sur des forums ou toute autre page pouvant vous guider vers la solution.
- Vous n'avez pas droit à la page UVA toolkit (<http://uvatoolkit.com/problemssolve.php>), à vous de générer vos cas de tests et de vérifier vos solutions.
- Tout document papier est autorisé dans une limite de 25 pages (police minimum 10).
- Vous n'avez pas le droit de réutiliser vos anciens codes, ni même d'aller les relire (sauf sur votre document papier évidemment).
- Il y a 8 problèmes, vous avez 4 heures pour en traiter un maximum.
- À la fin de la séance, vous devez envoyer par mail aux 3 TD-wo,men vos codes dans un tarball ayant le nom suivant : <login groupe>.tgz. Chaque problème devra être dans un fichier ayant pour nom le numéro du problème (l'extension dépend bien entendu du langage utilisé).
- Petit rappel : traitez en priorité les exercices faciles.

1 Division (10083)

Time limit: 3.000 seconds.

Given t , a , b positive integers not bigger than 2147483647, establish whether $\frac{(t^a-1)}{(t^b-1)}$ is an integer with less than 100 digits. Each line of input contains t , a , b . For each line of input print the formula followed by its value, or followed by "is not an integer with less than 100 digits", whichever is appropriate.

1.1 Sample input

```
2 9 3
2 3 2
21 42 7
123 911 1
```

1.2 Output for sample input

```
(2^9-1)/(2^3-1) 73
(2^3-1)/(2^2-1) is not an integer with less than 100 digits.
(21^42-1)/(21^7-1) 18952884496956715554550978627384117011154680106
(123^911-1)/(123^1-1) is not an integer with less than 100 digits.
```

2 Marbles on a tree (10672)

Time limit: 3.000 seconds.

n boxes are placed on the vertices of a rooted tree, which are numbered from 1 to n , $1 \leq n \leq 10000$. Each box is either empty or contains a number of marbles; the total number of marbles is n .

The task is to move the marbles such that each box contains exactly one marble. This is to be accomplished by a sequence of moves; each move consists of moving one marble to a box at an adjacent vertex. What is the minimum number of moves required to achieve the goal?

The input contains a number of cases. Each case starts with the number n followed by n lines. Each line contains at least three numbers which are: v the number of a vertex, followed by the number of marbles originally placed at vertex v followed by a number d which is the number of children of v , followed by d numbers giving the identities of the children of v .

The input is terminated by a case where $n = 0$ and this case should not be processed.

For each case in the input, output the smallest number of moves of marbles resulting in one marble at each vertex of the tree.

2.1 Sample input

```
9
1 2 3 2 3 4
2 1 0
3 0 2 5 6
4 1 3 7 8 9
5 3 0
6 0 0
7 0 0
8 2 0
9 0 0
9
1 0 3 2 3 4
2 0 0
3 0 2 5 6
4 9 3 7 8 9
5 0 0
6 0 0
7 0 0
8 0 0
9 0 0
9
1 0 3 2 3 4
2 9 0
3 0 2 5 6
4 0 3 7 8 9
5 0 0
6 0 0
7 0 0
8 0 0
9 0 0
0
```

2.2 Output for sample input

```
7
14
20
```

3 Combo Deal (10898)

Time limit: 3.000 seconds.

A fast food store offers a series of "combo meal deals" in addition to individually priced items. For example, the menu at the store may look like this:

Hamburger	\$3.49
Fries	\$0.99
Pop	\$1.09
Ice Cream	\$2.19
Value Meal (1 Hamburger, 1 Fries, 1 Pop)	\$4.79
Lovers-Only (2 Hamburgers, 2 Fries, 2 Pops, 1 Ice Cream)	\$9.99

Buying a combo is cheaper than buying its items individually.

A parent of many kids (or a coach of many students) face this recurring problem: I need to get, say, 9 hamburgers, 6 fries, and 8 pops. How do I fit this into the menu, using the combo deals optimally, so as to pay as little as possible? Note that I am a conservativist, so I don't buy more food than I need.

3.1 Input

The input contains several test cases, each of them with a menu and several orders.

1. Menu: Individual items, then combos.
 - Individual items: number of items $I \leq 6$, then their prices (at most \$10 each).
 - Combos: number of combos (at most 8), then for each combo, its composition as an I -tuple of quantities and its price.

Example: the sample input below encodes the menu above.

2. Orders: number of orders (at most 10), then for each order, an I -tuple of the wanted quantities. Each element in the tuples is at most 9.

All prices are integers in cents.

3.2 Output

For each order of each case, output the minimum payment in cents on its own line.

3.3 Sample Input

```
4 349 99 109 219
2
1 1 1 0 479
2 2 2 1 999\hfill
2
9 6 8 0
9 6 8 5
```

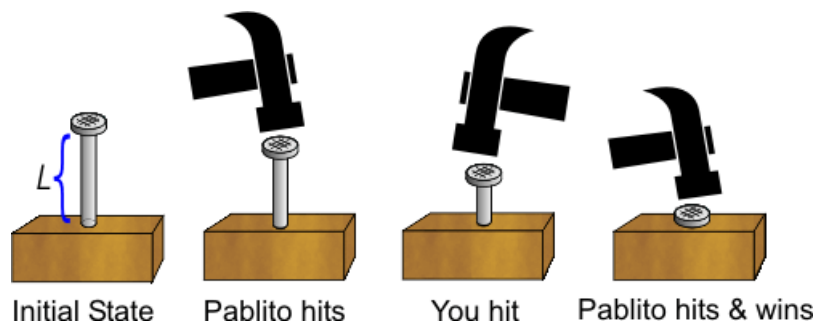
3.4 Sample Output

```
4139
4700
```

4 Pablito nailed a nail (10853)

Time limit: 3.000 seconds.

Pablito will give you a birdie if you can beat him in his favorite game: nailing the nail. This game is a pretty mixture of intelligence and brute force. There is a long nail, a bit inserted into a piece of wood. You and Pablito hit the nail in turns with a hammer. The winner is the first player who completely inserts the nail.



All measures are given in integer numbers. Depending on his/her strength and ability, each player can push the nail between a minimum, $Xmin$, and a maximum quantity, $Xmax$. However, if the current length of the nail is less than $Xmin$, the player can insert the nail completely and win.

Let's name the players A and B. The optimal strategy for player A (similarly for B) is defined as follows:

- a hit where A inserts the nail completely is an optimal hit for A,
- if (a) is not possible, a hit of A that leads to an optimal hit of A after all possible hits of B, is also an optimal hit of A,
- if neither (a) nor (b) are possible, A has no optimal hit, and will insert the nail any valid quantity.

Suppose player A always hits first, and both players use the optimal strategy for them. Who will win the game?

4.1 Input

The first line of the input contains an integer N , indicating the number of test cases.

Each test case is described in a single line, containing 5 integers: $L Amin Amax Bmin Bmax$, indicating the initial length of the nail, the minimum hit of A, the maximum hit of A, the minimum hit of B, and the maximum hit of B, respectively. Assume all numbers are between 1 and 2^30 , $Amin \leq Amax$, and $Bmin \leq Bmax$.

4.2 Output

For each test case, the output should consist of a line with a single letter, A or B, indicating the winner in that case.

4.3 Sample Input

```
4
4 5 7 1 20
5 1 3 1 3
5 2 2 1 3
1000 1 3 1 3
```

4.4 Sample Output

```
A
A
B
B
```

5 Equation Solver (533)

Time limit: 3.000 seconds.

Write a program that can solve linear equations with one variable.

5.1 Input Specifications

The input file will contain a number of equations, each one on a separate line. All equations are strings of less than 100 characters which strictly adhere to the following grammar (given in EBNF):

```
Equation  := Expression '=' Expression
Expression := Term { ('+' | '-') Term }
Term      := Factor { '*' Factor }
Factor    := Number | 'x' | '(' Expression ')
Number    := Digit | Digit Number
Digit     := '0' | '1' | ... | '9'
```

Although the grammar would allow to construct non-linear equations like $x*x=25$, we guarantee that all equations occurring in the input file will be linear in x . We further guarantee that all sub-expressions of an equation will be linear in x too. That means, there won't be test cases like

```
x*x-x*x+x=0
```

which is a linear equation but contains non-linear sub-expressions ($x*x$).

Note that all numbers occurring in the input are non-negative integers, while the solution for x is a real number.

5.2 Output Specifications

For each test case, print a line saying "Equation # i (where i is the number of the test case) and a line with one of the following answers:

- If the equation has no solution, print "No solution."
- If the equation has infinitely many solutions, print "Infinitely many solutions."
- If the equation has exactly one solution, print " $x = \text{solution}$ " where solution is replaced by the appropriate real number (printed to six decimals).

Print a blank line after each test case, but the last one.

5.3 Sample Input

```
x+x+x=10
4*x+2=19
3*x=3*x+1+2+3
(42-6*7)*x=2*5-10
```

5.4 Sample output

```
Equation #1
x = 3.333333

Equation #2
x = 4.250000

Equation #3
No solution.

Equation #4
Infinitely many solutions.
```

6 Gleaming the Cubes (737)

Time limit: 3.000 seconds.

As chief engineer of the Starship Interprize, the task of repairing the hyperstellar, cubic, transwarped-out software has fallen on your shoulders. Simply put, you must compute the volume of the intersection of anywhere from 2 to 1000 cubes.

6.1 Input and Output

The input data file consists of several sets of cubes for which the volume of their intersections must be computed. The first line of the data file contains a number (from 2 to 1000) which indicates the number of cubes which follow, one cube per line. Each line which describes a cube contains four integers. The first three integers are the x , y and z coordinates of the corner of a cube, and the fourth integer is the positive distance which the cube extends in each of the three directions (parallel to the x , y , and z axes) from that corner.

Following the data for the first set of cubes will be a number which indicates how many cubes are in a second set, followed by the cube descriptions for the second set, again one per line. Following this will be a third set, and so on. Your program should continue to process sets of cubes, outputting the volume of their intersections to the output file, one set per line, until a zero is read for the number of cubes.

Note that the data file will always contain at least one set of cubes, and every set will contain at least 2 and at most 1000 cubes. For any given set of cubes, the volume of their intersections will not exceed 1,000,000 units.

6.2 Sample Input

```
2
0 0 0 10
9 1 1 5
3
0 0 0 10
9 1 1 5
8 2 2 3
0
```

6.3 Sample output

```
25
9
```

7 XYZZY (10557)

Time limit: 3.000 seconds.

ADVENT: `\advent\`, n.

The prototypical computer adventure game, first designed by Will Crowther on the PDP-10 in the mid-1970s as an attempt at computer-refereed fantasy gaming, and expanded into a puzzle-oriented game by Don Woods at Stanford in 1976. (Woods had been one of the authors of INTERCAL.) Now better known as Adventure or Colossal Cave Adventure, but the TOPS-10 operating system permitted only six-letter filenames in uppercase. See also vadding, Zork, and Infocom.

It has recently been discovered how to run open-source software on the Y-Crate gaming device. A number of enterprising designers have developed Advent-style games for deployment on the Y-Crate. Your job is to test a number of these designs to see which are winnable.

Each game consists of a set of up to 100 rooms. One of the rooms is the start and one of the rooms is the finish. Each room has an energy value between -100 and $+100$. One-way doorways interconnect pairs of rooms.

The player begins in the start room with 100 energy points. She may pass through any doorway that connects the room she is in to another room, thus entering the other room. The energy value of this room is added to the player's energy. This process continues until she wins by entering the finish room or dies by running out of energy (or quits in frustration). During her adventure the player may enter the same room several times, receiving its energy each time.

The input consists of several test cases. Each test case begins with n , the number of rooms. The rooms are numbered from 1 (the start room) to n (the finish room). Input for the n rooms follows. The input for each room consists of one or more lines containing:

- the energy value for room i
- the number of doorways leaving room i
- a list of the rooms that are reachable by the doorways leaving room i

The start and finish rooms will always have energy level 0. A line containing -1 follows the last test case.

In one line for each case, output "winnable" if it is possible for the player to win, otherwise output "hopeless".

7.1 Sample Input

```
5
0 1 2
-60 1 3
-60 1 4
20 1 5
0 0
5
0 1 2
20 1 3
-60 1 4
-60 1 5
0 0
5
0 1 2
21 1 3
-60 1 4
-60 1 5
0 0
5
0 1 2
20 2 1 3
-60 1 4
-60 1 5
0 0
-1
```

7.2 Sample output

```
hopeless
hopeless
winnable
winnable
```

8 Kindergarten Counting Game (494)

Time limit: 3.000 seconds.

Everybody sit down in a circle. Ok. Listen to me carefully.

“Wooooooo, you scewwy wabbit!”

Now, could someone tell me how many words I just said?

8.1 Input and Output

Input to your program will consist of a series of lines, each line containing multiple words (at least one). A “word” is defined as a consecutive sequence of letters (upper and/or lower case).

Your program should output a word count for each line of input. Each word count should be printed on a separate line.

8.2 Sample Input

Meep Meep!

I tot I taw a putty tat.

I did! I did! I did taw a putty tat.

Shsssssssssh ... I am hunting wabbits. Heh Heh Heh Heh ...

8.3 Sample Output

2

7

10

9