17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem A — Yellow Balloon
Find x

Problem Description

You have to implement a simple equation solver to evaluate expressions consisting of numbers, variables, and addition/subtraction operators.

In the past, this type of problem was called a "find x" problem, however, an unintended consequence of ubiquitous touch interfaces resulted in people fruitlessly pointing at the "x" on the screen, so we now require the unnecessarily complex syntax specified below to avoid this all-too-common misunderstanding.

Input

Your input consists of an arbitrary number of records, but no more than 100. Each record starts with a line containing a single integer n, with $3 \leq n \leq 32$, denoting the number of symbols to follow, followed on the next line by n single-character symbols, separated by a single space.

The symbols are selected from the following classes:

NUM : Numbers, the characters "0" through "9". All numbers specified in the input will be single-digit values;

EQ : Equals, the character "=";

VAR : Variables, the characters "a" through "z" (only lowercase), one of which will appear exactly once in any given equation;

OP : Operators, the characters "+" and "−". An operator may appear as the first symbol in an equation, or the first symbol after an EQ symbol — in this case it acts as a unary operator, which negates the NUM or VAR immediately to its right if the OP is "−", and does nothing if the OP is "+".

Valid equation syntax is

```
[OP] (NUM | VAR) (OP (NUM | VAR))* EQ [OP] (NUM | VAR) (OP (NUM | VAR))*
```

with the constraint that any given equation will contain only one instance of `VAR` (only one letter, appearing at only one place). In the syntax, parentheses denote grouping, the Or symbol "`|`" denotes a choice between its left and right operands, the square brackets "`[]`" indicate zero or one occurrences of the enclosed term), and the "`*`" symbol indicates zero or more occurrences of the term to its left.

The end of input is indicated by a line containing only the value −1.

Output

For each input record, output

```
VAR = answer
```

where `VAR` is the symbol used in the input equation, and `answer` is the solution to the equation.

Sample input

```
7
4 - 5 - x = 2
8
- 4 - 5 - y = 2
9
+ 4 - 5 - z = + 2
8
+ 4 - 5 = a + 2
9
+ 4 - 5 = - d + 2
-1
```

Sample output

```
x = -3
y = -11
z = -3
a = -3
d = 3
```

Time limit

1 second

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem B — White Balloon
Axles

Problem Description

You are running a machine shop producing custom axles for radio-controlled cars. These axles can be manufactured from either steel or stainless steel, with the stainless steel parts usually priced differently from plain steel.

You have two CNC lathes: one is configured to produce stainless steel parts, and the other is configured to produce steel parts. In a given production run, a specific part design can only be assigned to one of the two machines, i.e., the same part design can be manufactured in either steel or stainless steel, but not both.

The parts are produced from solid rods of material; today your stock levels are pretty low and you only have one length of steel rod, and one length of stainless steel rod.

Part designs are selected from a library, where each part is listed according to the length of stock material required to produce it, as well as the current profit yield of the part in both steel and stainless steel versions.

You must select a subset of parts from your library, and decide how many copies of those parts to produce, as well as whether a given part should be produced in steel or stainless steel, in a way that maximises your profit given your available material stock.

Input

Your input consists of an arbitrary number of records, but no more than 30. Each record starts with an integer n, with $2 \leq n \leq 14$, denoting the number of parts available in your library.

The next n lines each provides three positive integer values,

`m p s`

where `m` denotes the length of stock material (in millimetres) consumed when producing one of these parts, `p` denotes the profit made when manufacturing the part from steel (in Rands), and `s` denotes the profit made when manufacturing the part in stainless steel, subject to the constraints $1 \leq \texttt{m} \leq 1500$ and $1 \leq \texttt{p}, \texttt{s} \leq 1000$.

After these n lines follows another line containing two positive integer values,

`q r`

where `q` denotes the length of steel stock you have available (in millimetres), and `r` denotes the length of stainless steel stock you have available, subject to $1 \leq \texttt{q} \leq 1000$ and $1 \leq \texttt{r} \leq 1200$.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

```
u
```

where `u` denotes the maximum profit to be made with the available stock material.

Sample input

```
3
10 200 300
20 300 200
10 200 600
200 100
13
280 306 484
185 335 212
38 110 40
96 236 124
256 478 279
256 301 480
249 292 445
258 299 437
281 287 472
44 62 142
349 369 593
258 271 466
208 406 243
998 1123
-1
```

Sample output

```
10000
6410
```

Time limit

10 seconds

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem C — Red Balloon
Tall orders

Problem Description

You have been charged with evaluating the safety of power cables passing over train tracks. Owing to the recent purchase of the new Macho 10000 trains, the questions has been raised as to whether the trains are too tall to safely pass under existing power cable infrastructure.

The new Macho 10000 train is 4.1 m tall; the power engineers claim that the power lines have been designed for a maximum train height of 4.05 m, with a safety gap of 150 mm for thermal expansion in the cables. Fortunately the data required to test this claim is available: you have a complete database of the critical dimensions of all power line infrastructure passing over train tracks. In particular, you have the following dimensions (see Figure 1):

- The distance between the two posts supporting the power cable, d. You may safely assume that the train track is exactly halfway between the two posts.
- The height of the top of the posts above the track, p. You may safely assume that the cable is attached right at the top of the post.

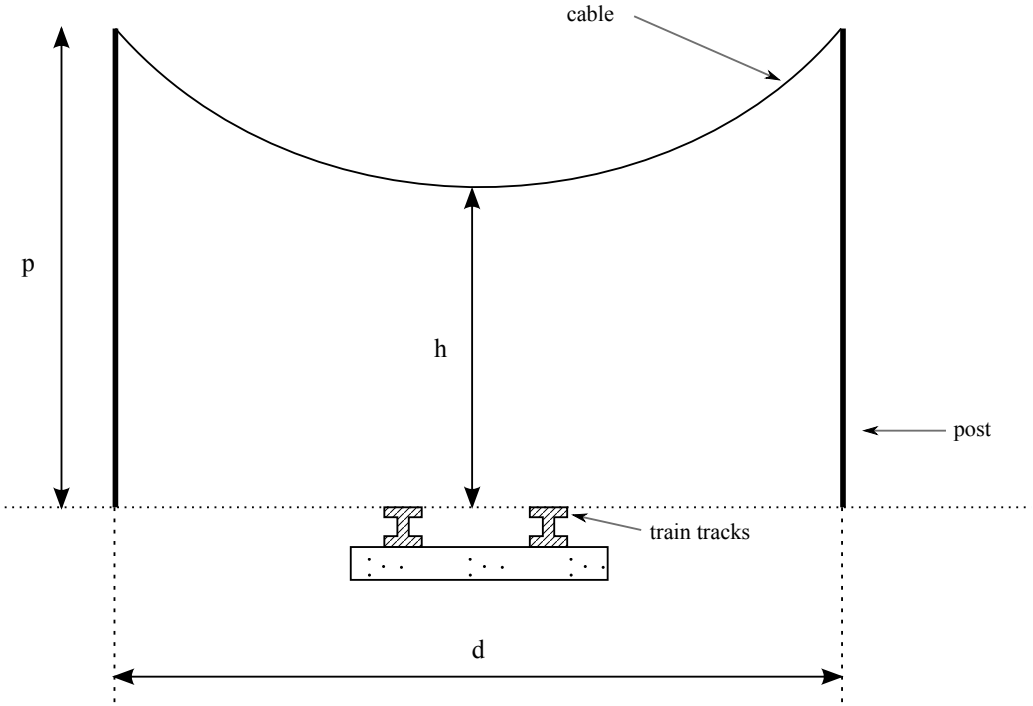

Figure 1: Power line dimensions

A cable hanging between poles is known to assume a specific shape called a *catenary*. This shape is

described by the formula

$$f(s) = a \cosh\left(\frac{s}{a}\right),$$

where $\frac{-d}{2} \leq s \leq \frac{d}{2}$ denotes a position along the cable, as measured on the ground, and

$$\cosh(x) = \frac{e^x + e^{-x}}{2}$$

A given configuration (specific d and p values) can be said to be safe if the lowest point along the cable is at a height of 4.2 m above the track. By modeling the cable as an infinitely thin thread, the function $f(s)$ can be used to determine the maximum length of cable that will be safe.

Input

Your input consists of an arbitrary number of records, but no more than 100.

Each record comprises two real numbers,

`p d`

where $4.3 \leq$ `p` ≤ 10 denotes the height of the top of the posts above the tracks (in metres), and $6 \leq$ `d` ≤ 30 denotes the distance between the two posts, also given in metres (see Figure 1).

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

`L`

where `L` denotes the maximum length of cable that may be used while keeping the lowest point along the cable 4.2 m or more above the tracks.

The value `L` should be given in metres, and should be truncated (not rounded) to three places after the decimal point.

Sample input

```
6 12
6.210381 10.184095
-1
```

Sample output

```
12.692
11.175
```

Time limit

30 seconds

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem D — Pink Balloon
Bouncy bounce

Problem Description

While engaging in the nefarious act of doomsday device building, you have run into a practical problem. A signaling laser has to be passed through a chamber with mirrored walls, but owing to the other functions of the chamber, all of them nefarious, the laser beam's path can become quite convoluted. The chamber possesses both an entry port, through which the laser beam enters, and an exit port, through which the laser beam must leave the chamber. These ports have a fixed location (see Figure 1), and the origin of the laser beam is also fixed.

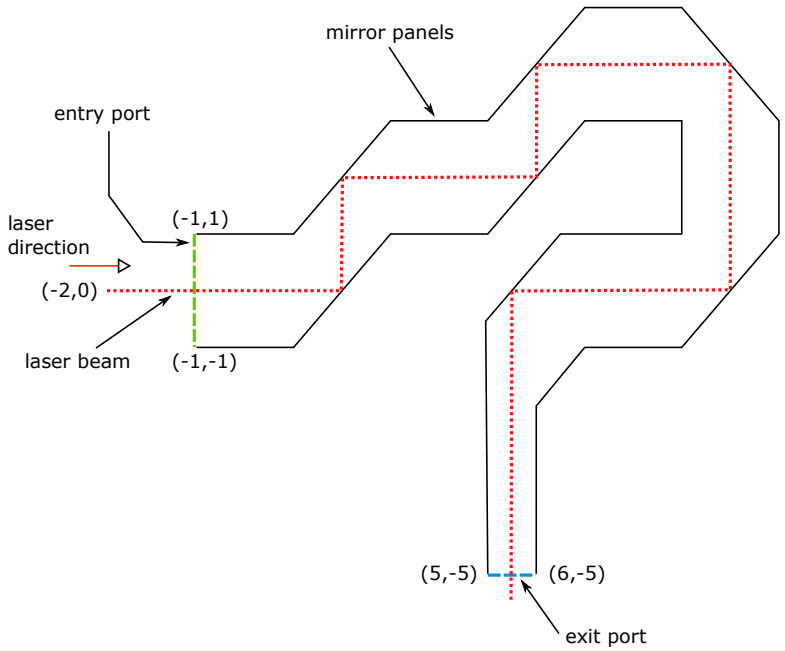

Figure 1: Mirrored chamber with laser beam. Note that the initial laser beam direction indicated is specific to this example.

Given a 2D representation of the chamber, and the initial orientation of the laser beam, you must determine whether the beam will reach the exit port in 10 or fewer bounces off the mirrored walls. The mirrors reflect the laser beam according to the *law of reflection*, such that the angle of reflection is equal to the angle of incidence.

The world is not going to hold itself hostage, so you better get cracking on that doomsday device.

Input

Your input consists of an arbitrary number of records, but no more than 30. Each record describes the shape of the chamber as a pair of polygonal chains. The first polygonal chain runs from the lower vertex of the entry port, $(-1, -1)$, to the left vertex of the exit port, $(5, -5)$. The second runs from the right vertex of the exit port, $(6, -5)$, to the upper vertex of the entry port, $(-1, 1)$.

Each input record starts with the value n, with $2 \leq n \leq 32$, denoting the number of vertices in the first polygonal chain, followed by n input lines each containing a pair of real numbers representing the x and y coordinates of a vertex along the chain. The vertices are specified in a counterclockwise order, and always start with $(-1, -1)$ and end with $(5, -5)$.

The next line following these n pairs of coordinates contains the integer m, with $2 \leq m \leq 32$, denoting the number of vertices in the second polygonal chain. This is followed by m input lines each containing a pair of real numbers representing the x and y coordinates of a vertex along the second polygonal chain. The vertices are specified in a counterclockwise order, and always start with $(6, -5)$ and end with $(-1, 1)$.

After these m lines follows a pair of real numbers `px py` indicating a second point through which the laser beam passes; the origin of the laser beam is fixed at $(-2, 0)$.

The input data is guaranteed to never produce a ray that will intersect a vertex. The beam is also guaranteed to enter through the entry port. The coordinates are limited to the rectangular region $[-1, -15]$ to $[15, 15]$. Lastly, the polygonal chains never intersect themselves, each other, or the entry/exit ports.

The end of input is indicated by a line containing only the value -1.

Output

Follow the path of this beam by reflecting it when it encounters the polygonal chains that define the mirror chamber. The beam exits the mirror chamber if it passes through the virtual line segment between $(5, -5)$ and $(6, -5)$. In that case, output

```
beam exited
```

If the beam fails to pass through the exit port after at most 10 reflections, or if it leaves through the entry port, output

```
beam failed to exit
```

Sample input

```
2
-1 -1
5 -5
4
6 -5
10 1
6 5
-1 1
-1 0
3
-1 -1
1 -1
5 -5
5
6 -5
11 0
5.5 5.5
1 1
-1 1
-1 0.3
-1
```

Sample output

```
beam failed to exit
beam exited
```

Time limit

1 second

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem E — Purple Balloon Trampolines

Problem Description

You are being chased by police after repossessing a laser for a nefarious doomsday device. To escape, you need to cross a field of trampolines. Fortunately for you, the trampolines are arranged in a regular grid, but unfortunately they are placed on platforms of different heights.

Your task is to find the smallest number of jumps required to cross the field, entering at the north-west corner and exiting at the south-east corner. The height of each trampoline is given in metres, as illustrated in Figure 1.

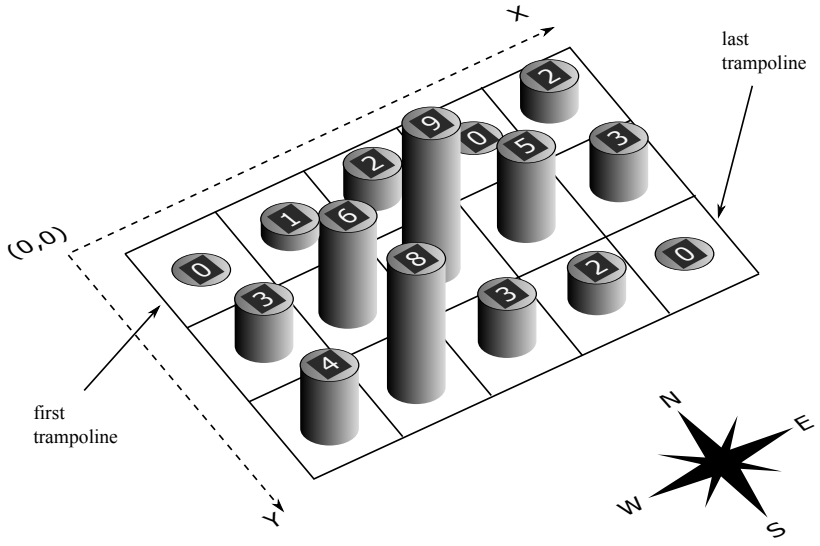

Figure 1: A grid of trampolines. The trampolines are supported on platforms of varying height.

When you drop onto a trampoline, you can use your knees to exert some control over the height of your next jump. The high-point of your next jump can be up to 1 m higher or lower than your previous one, provided that you rise at least 1 m and no more than 4 m from the trampoline. You must also ensure that the high-point of your jump is at least 1 m above the trampoline on which you will land (so that you can bounce), but no more than 5 m (otherwise you will break a leg).

You also have some horizontal control. You can always land on the same trampoline from which you took off: call this a "non-moving" jump. You can also jump from a trampoline to a neighbouring one (north, south, east or west) — a "moving" jump. However, you can only do two moving jumps in a row if they are in the same direction; you need a non-moving jump to change direction.

It is also possible to make a double jump, where you skip over a trampoline. This can be done under the following conditions (in addition to those already mentioned):

- The jumped-from and jumped-to trampolines must have the same height.
- The trampoline in between must be at the same height or lower than the other two trampolines.
- The previous jump must be a moving jump, in the same direction as the double jump (double jumps are also moving jumps).

Your first jump is a moving jump onto the north-western trampoline, dropping 1 m onto it, from either the north or west (your choice). Your final jump is a moving jump off the south-eastern trampoline, rising 1 m from it, either to the south or east (again, your choice).

It is guaranteed that it will be possible to cross the field in all test cases.

To illustrate, the following is a solution to the grid shown in Figure 1. Coordinates represent trampolines, from $(0, 0)$ to $(4, 2)$, and heights represent the height of the top of each jump: – 1 m – (0, 0) – 2 m – (1, 0) – 3 m – (2, 0) – 3 m – (4, 0) – 4 m – (4, 0) – 4 m – (4, 1) – 4 m – (4, 2) – 3 m – (4, 2) – 2 m – (4, 2) – 1 m – .

Input

The input consists of an arbitrary number of records, but no more than 20. Each record starts with a line containing integers X and Y, $3 \leq X, Y \leq 100$, the width and height of the field. This is followed by Y lines of X space-separated integers, giving the heights of the trampolines in metres. The lines run north to south and each line runs west to east. All heights are in the range 0 – 999, inclusive.

Input is terminated by a line containing only -1.

Output

For each record output the minimum possible number of jumps required to cross the field, including the jumps to get onto the first trampoline and off of the last and back onto solid ground.

Sample input

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

Sample output

```
10
11
```

Time limit

10 seconds

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem F — Blue Balloon
Getting your priorities straight

Problem Description

After implementing a newfangled type of priority queue data structure, you now have to write a unit test to ensure that your priority queue is working as advertised. The objective of the priority queue is to arrange a set of elements into non-descending order, such that for the non-negative integers x_i, with $1 \leq i \leq n$, we have that

$$x_1 \leq x_2 \leq x_3 \leq \ldots \leq x_{n-1} \leq x_n$$

Your mission, should you choose to accept, is to test the output of the priority queue to ensure that the numbers are arranged in non-descending order.

Input

Your input consists of an arbitrary number of records, but no more than 250. Each record starts with a line containing a single integer n, with $2 \leq n \leq 50$, denoting the number of values to follow. The next line contains n integers x_i separated by spaces, subject to $1 \leq x_i \leq 100$.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

`yes`

if the numbers x_i are in non-descending order. If not, your program must output

`no`

Sample input

```
5
1 2 3 4 5
7
1 2 4 3 12 99 100
2
2 1
6
2 2 2 2 2 2
-1
```

Sample output

```
yes
no
no
yes
```

Time limit

1 second

17^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

24 October 2015

Problem G — Orange Balloon Pay Day

Problem Description

You work for a mining company deep in the asteroid belt. Due to recent political instability in the outer system, employees have become distrustful of the Banking System and are demanding to be paid in cash. In order to do so you need a specific quantity of each denomination of currency.

Your job is to write a program which will calculate the quantity of each denomination needed to pay every employee their due — or face a bloody rebellion. Everyone expects to receive their pay using the largest denominations possible.

Input

Your input consists of an arbitrary number of records, but no more than 100. Each record starts with a line containing a single integer m, with $1 \leq m \leq 100$, denoting the number of salaries to follow. The next line contains m integers s_i separated by spaces, subject to $1 \leq s_i \leq 500\,000$, representing the salaries of your m workers.

The denominations are the fixed set of coins with values 30030 2310 210 30 6 2 1.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output the quantity of each denomination required. Keep in mind that employees want as many coins of the largest denomination possible, followed by the maximum of the next largest denomination possible, and so on until the smallest denomination is reached.

You must output a line formatted as

```
denomination = quantity
```

for each denomination (even if the quantity is zero), in the order they were specified. After each record, output a blank line.

Sample input

```
5
162 161 163 164 165
2
203236 300260
-1
```

Sample output

```
30030 = 0
2310 = 0
210 = 0
30 = 25
6 = 9
2 = 4
1 = 3

30030 = 15
2310 = 21
210 = 20
30 = 10
6 = 5
2 = 3
1 = 0
```

Time limit

5 seconds