The 8th Annual University of Akron Programming Competition

presented by

The University of Akron Computer Science Department

Association for Computing Machinery Student Chapter

April 14, 2018

Rules:

1. There are six questions to be completed in four hours.
2. C, C++, and Java are the only languages available.
3. Data is read from Standard Input and output is sent to Standard Output. Do not prompt for input values in the code you submit to be judged. Do not attempt to read from or write to any files.
4. All programs are submitted as source code only. Submitting compiled information (binary, .class) will result in a Compilation Error penalty.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation).
6. The input to all problems will consist of multiple test cases. The Sample Input listed on the problem description is not intended to be a comprehensive input set. The input is guaranteed to adhere to the descriptions in each problem; you do not need to check for invalid input.
7. The output to all problems should conform to the format shown in the Sample Output for that problem, including but not limited to capitalization, whitespace, etc.
8. Use of personal electronics during the competition is cause for disqualification. Cell phones must be turned all the way off (not silent mode, not airplane mode, etc.). You may use any written notes, books, or reference materials you bring with you.
9. Programming style is not considered in this contest. You can code in any style or with any level of documentation your team prefers.
10. All communication with the judges will be handled through PC^2. All judges decisions are final.
11. Source code is compiled on the command line, C and C++ will be compiled with GNU G++ (`g++ sourcefile.cpp`) and Java with the standard Java RE (`javac sourcefile.java`).

A: Roman Reigns

Kevin needs help converting Roman numerals into the Arabic numerals (base 10 integer numbers). The rules for converting are simple. All the numbers are added up from left to right. The one exception is if a larger value number follows a smaller value number, the smaller value is subtracted from the larger value. The values for Roman numbers are:

- I: 1
- V: 5
- X: 10
- L: 50
- C: 100
- D: 500
- M: 1000

Input

The first line of input consists of one Roman numeral.

Output

Output one integer, the value of the Roman numeral as an Arabic numeral. The output is known to be between 1 and 4999.

Sample Input	Sample Output
`XLIX`	`49`

Sample Input	Sample Output
`CMXCIX`	`999`

Sample Input	Sample Output
`MCCXXXIV`	`1234`

B: Test Memorization

Janet needs to memorize a bunch of different words for her test. The professor informed her that every word would be its own question. Obviously, Janet decided that she wanted to memorize as many cards as possible to maximize her score. Unfortunately, there are a *lot* of cards that she needs to memorize so figuring out which ones to study first by hand is very cumbersome.

Each card has one word on it that starts with an ASCII lowercase alpha character (`'a'-'z'`) and continues with either an ASCII lowercase alpha character or a numeral (`'0'-'9'`). Janet was able to write a small program herself to label the cards with how many seconds she thinks each one will take to memorize. She also knows how many seconds she has available to study before the test. Help Janet out by telling her which cards she should study in order to maximize her score on the test!

Input

The first line of input consists of two integers, n ($1 \leq n \leq 10^4$), the number of cards, and k ($1 \leq k \leq 10^{10}$), the total number of seconds available to study. The next n lines consist of a string s_i, the name of the ith card, and an integer, t_i ($1 \leq t_i \leq k$), the time in seconds needed to memorize the ith card.

Output

Output one integer, the maximum number of cards that can be memorized in the time available to study.

Sample Input

```
5 10
rememberful 8
joykingly 4
hep 2
mayhaps 2
belive 5
```

Sample Output

```
3
```

Sample Input

```
4 10
heppily 8
mutin 2
terseful 9
googlex 8
```

Sample Output

```
2
```

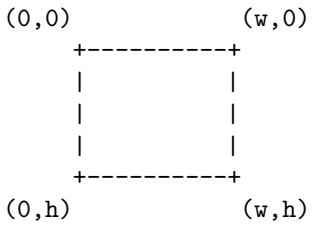
C: Spans

Mikhail has been hard at work on his calculator 3D engine, Ecks3D. His goal is to write the fastest software renderer ever written. The problem is, 3D rendering is hard stuff. Luckily, he's already written a lot of the mathy stuff and just needs help with one final step: occlusion culling.

In a 3D scene, you have lots of polygons. When you draw these on the screen, the closer polygons cover up the polygons that are futher away (since humans have not yet evolved the ability to see through walls). This is known as occlusion culling, since closer polygons "occlude" the ones that are futher away. Your task is to take a set of polygons and draw the correctly occluded 3D image. To help you do this, every polygon has been given a depth value, which represents how close to the viewer a polygon is. Polygons that have a larger depth value are closer to the viewer and should occlude polygons that have a smaller depth value. It is guaranteed that no two polygons have the same depth value. Can you help Mikhail achieve his dream?

Polygons are represented as a set of spans, which are horizontal slices of a polygon. The span defines, for a given y value, the horizontal range of pixels that are in the polygon. For example, if a span has a y of 10, left of 5, and right of 7, that indicates the pixels (5, 10), (6, 10), and (7, 10) are all in the polygon. All of the spans in a given polygon have the same color (represented as an ASCII character) and depth value.

Of course, it's impossible to draw pixels if we don't use the proper coordinate system! The screen in Mikhail's engine uses a coordinate system where the x-axis increases to the left and the y-axis increases down.



In the example below, we want to draw a rectangle consisting of '.'s with a depth of 5. In addition, we want to draw circle of '#'s with a depth of 10. Because the circle is closer, it covers up parts of the rectangle.

Input

The first line of input consists of two integers, W ($1 \leq W \leq 320$), the width of the screen in pixels, and H ($1 \leq H \leq 240$), the height of the screen in pixels. The next line has one integer P ($1 \leq P \leq 10000$), the number of polygons in the scene.

Next is the description of each polygon. The ith polygon begins with one line that consists of a character $C[i]$, the color of the polygon, an integer $D[i]$, the depth value of the polygon, and an integer $N[i]$, the number of spans in the polygon. Following this line are $N[i]$ total lines describing the spans in the polygon. The j'th span consists of three integers: $LEFT[j]$, $RIGHT[j]$ ($-10^9 \leq LEFT[j] \leq RIGHT[j] \leq 10^9$) and $Y[j]$ ($-10^9 \leq Y[j] \leq 10^9$). The number of spans across all of the polygons is guaranteed to be no more than 10^6.

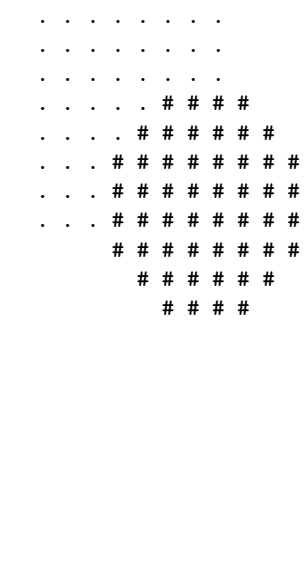
Output

Output only the pixels that are on the screen (remember some pixels may be off the screen!) with a space in between each pixel. A space is put between each pixel because most monspace fonts are taller than they are wide. If the pixel wasn't drawn, output a space instead of the color. Thus, you should print a grid that is $W \times 2$ by H characters.

Sample Input

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

Sample Output



D: Truffula Trees

Kevin, whose name is definitely not Justin, has heard all of this talk of "graphs" and "trees" in computer science and thought it would be neat if he could create his own weird Dr. Seuss hybrid. That way, people would respect him enough to get his name right. So, that's exactly what he did - it's a tree that looks like a line graph. He needed a name though, and decided to go with "Truffula trees", a name shamelessly ripped off from *The Lorax*. After all, every tree was destroyed in the book by the Oncler, so they can't possibly sue for copyright infringement, *right*? (Someone didn't read until the end of the book!)

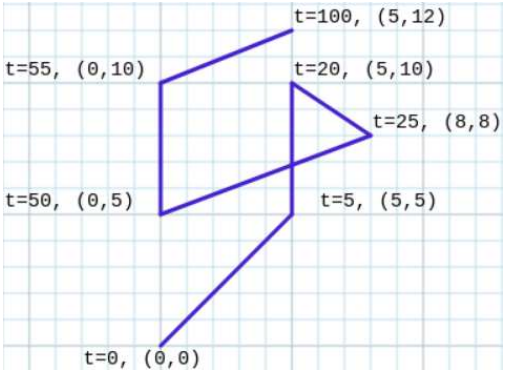
After creating one, he planted the seed in the ground and watched it grow. It was incredible! But also very strange. You see, Truffula trees grow in a very unique way. It begins growing at the root where the seed was planted, the coordinate $(0, 0)$. The tree is made of many line segments which are joined by joints. The tree only likes to grow in straight lines, and the rate at which it grows stays constant for the whole segment. But it can't keep growing in the same direction forever! So after a while, it makes a new joint and starts growing in a new direction. Strangely enough, the length of the segment has nothing to do with how long it took that segment to grow. Justin *er* - Kevin, decided to study his incredible creation in more depth by recording the position of each joint and the time at which the joint was formed. To be sure, he also added records for the root and the top of the tree.

For his study, Kevin wants to know where the top of the tree is at a given time. Being straight outta, well, Seuss, the tree can have loops and all sorts of crazy bends! There's a problem though. Knowing the position at some times is easy. For example, at $t = 0$ the tree is always at $(0, 0)$. And if the time happens to coincide with the time at a joint, then it's just whatever position he recorded. But what happens if the time is between two joints? Luckily, the segments grow in straight lines so we can use our happy formula friend, 2D linear interpolation, to answer his questions!

Recall that 2D linear interpolation takes three parameters: a start point (x_1, y_1), an end point (x_2, y_2), and t $(0 \leq t \leq 1)$, a parameter that describes the percentage of how far along the line you are. When $t = 0$, it will give the start point (x_1, y_1). When $t = 1$, it will give the end point (x_2, y_2). When $t = 0.5$, it will give the midpoint, etc. The formulas are as follows:

$$x(t) = x_1 + t \times (x_2 - x_1)$$
$$y(t) = y_1 + t \times (y_2 - y_1)$$

The point $(x(t), y(t))$ will tell you the coordinate on the line segment for a given t. For reference, here is a picture Kevin took of the tree from the first input:



Can you help Kevin conduct his research so that people get his name right?

Input

The first line of input consists of an integer N ($1 \leq N \leq 10^5$), the number of joints in the tree (this includes the root and top of the tree). Following are N total lines. The ith line has three integers: $TIME[i]$ ($0 \leq TIME[i] \leq 10^9$), the time at which the ith joint was formed, $X[i]$ ($-10^9 \leq X[i] \leq 10^9$), the x-coordinate of the joint, and $Y[i]$ ($0 \leq Y[i] \leq 10^9$), the y-coordinate of the joint. Note that $TIME[1] < TIME[2] < ... < TIME[N]$.

The next line has one integer, M ($1 \leq M \leq 10^5$), the number of queries that Kevin is going to make. Following are M total lines. The jth line describes a query which consists of one real number, $Q[j]$ ($0 \leq Q[j] \leq TIME[N]$), the time at which Kevinn wants to query the position of the top of the tree.

Output

For each query, output the x position and y position of the top of the tree at the give time on a new line, separated by a space. Your answer is considered correct if the absolute error does not exceed 10^{-6}.

Sample Input	**Sample Output**
7 0 0 0 5 5 5 20 5 10 25 8 8 50 0 5 55 0 10 100 5 12 3 55 4 11	0 10 4 4 5 7

Sample Input	**Sample Output**
6 0 0 0 2 4 4 3 1 0 8 3 0 11 3 0 16 6 4 5 14 4 5 0 6	4.8 2.4 1.4 0 1.8 0 0 0 2.2 0

E: Billy's Graph

Billy is a big fan of Graph Theory. When I say big, I mean he likes to invent types of graphs after himself because he wants to be a famous researcher in Graph Theory. Billy has come up with his latest and greatest invention of a type of graph and wants to share it with you. He calls these new graphs, Billy Graphs, they are indiciated by every edge sharing at least one node.

Billy is pretty good at Graph Theory and other mathematical disciplines but doesn't really care for programming. He thinks its a lesser science. Yet, here he is, asking you for help! Billy wants a program to tell him if an abritrary graph is a Billy Graph or not and so he's come to you for help.

Input

The first line of input consists of two integers, n ($3 \leq n \leq 1000$), the number of nodes, and m ($2 \leq m \leq n^2$), the number of edges. The next m lines consists of two integers, a and b, representing an edge between nodes a and b.

Output

Output either `YES` if the graph is a Billy Graph, or `NO` otherwise.

Sample Input	Sample Output
6 5 1 6 2 6 3 6 4 6 5 6	YES

Sample Input	Sample Output
5 5 1 2 2 3 3 4 2 4 4 5	NO

F: Peter Rabbit

Peter Rabbit, like other rabbits, needs to eat. His favorite food to eat is carrots. As Peter notices a carrot, he eats the carrot and searches for more carrots. He can hop up, down, left, and right on safe places of the ground to get to other carrots. It's very important for Peter Rabbit to avoid rabbit holes in the ground because if he falls into one, he cannot get out. Help Peter Rabbit collect all the carrots on the map while he avoids falling in any holes and avoids falling off the map.

Input

The first line contains two integers, x and y ($1 \leq x, y \leq 10$) where x is the length of the map and y is the width of the map.

The next x lines contains y characters. The possible characters are as follows:

- R - The starting postion of Peter Rabbit
- G - A safe piece of Ground for Peter Rabbit to hop to
- H - A Hole that Peter Rabbit must avoid
- C - A Carrot that Peter Rabbit wants to hop to

As Peter Rabbit consumes a Carrot that he lands on, the Carrot C then becomes ground G. It is known that Peter Rabbit is able to hop to any Carrot meaning no Carrot is isolated on an island.

Output

Output any valid sequence of hops up U, down D, left L, and right R so that Peter Rabbit collects all the carrots. Make sure that Peter Rabbit does not fall off the map and does not fall in a rabbit hole. The last move Peter Rabbit makes must land on a carrot.

Sample Input	**Sample Output**
`1 5` `R C C G C`	`RRRR`

Sample Input	**Sample Output**
`1 3` `C R C`	`LRR`

Sample Input	**Sample Output**

`2 5`
`C C G C C`
`C H H H R`
	`ULLLLD`