ANZAC Contest

August 22, 2020

Contest Problems

Problem set contains 11 problems over 24 pages. Compiled by Hossam Elgindy and Tim French.

This page is intentionally left (almost) blank.

Problem A
Majority

The votes are in! Mathematicians world-wide have been polled, and each has chosen their favorite number between 1 and 1000. Your goal is to tally the votes and determine what the most popular number is.

If there is a tie for the greatest number of votes, choose the smallest number with that many votes.

Input

For each test case, there will be a single line giving the number of votes V , $1 \leq V \leq 1000$. Following that line will be V lines, each with a single integer vote between 1 and 1000.

Output

For each test case, output the smallest number that recieved the greatest number of votes.

Sample Input 1

```
4
7
99
99
7
```

Sample Output 1

```
7
```

Sample Input 2

```
5
11
12
13
14
15
```

Sample Output 2

```
11
```

This page is intentionally left (almost) blank.

Problem B
Triangles

Determine if it is possible to produce two triangles of given side lengths, by cutting some rectangle with a single line segment, and freely rotating and flipping the resulting pieces.

Input

The input consists of two lines. The first line contains three space-separated positive integers, indicating the desired side lengths of the first triangle. Similarly, the second line contains three space-separated positive integers, denoting the desired side lengths of the second triangle. It is guaranteed that the side lengths produce valid triangles. All side lengths are less than or equal to 100.

Output

Print, on a single line, whether there exists a rectangle which could have been cut to form triangles of the given side lengths. If such a rectangle exists, print `YES`. Otherwise, print `NO`.

Sample Input 1

```
3 4 6
4 6 3
```

Sample Output 1

```
NO
```

Sample Input 2

```
39 52 65
25 60 65
```

Sample Output 2

```
NO
```

This page is intentionally left (almost) blank.

Problem C
Buggy Robot

You are trying to program a robot to navigate through a 2-dimensional maze and find the exit.

The maze can be represented as a grid with n rows and m columns. Some grid cells have obstacles that the robot cannot pass. The other cells are empty, which the robot can freely pass. Exactly one of the empty cells in the grid is marked as the exit, and the robot will exit the maze immediately once it reaches there.

You can program the robot by sending it a command string. A command string consists of characters `L`, `U`, `R`, `D`, corresponding to the directions left, up, right, down, respectively. The robot will then start executing the commands, by moving to an adjacent cell in the directions specified by the command string. If the robot would run into an obstacle or off the edge of the grid, it will ignore the command, but it will continue on to remaining commands. The robot will also ignore all commands after reaching the exit cell.

Your friend sent you a draft of a command string, but you quickly realize that the command string will not necessarily take the robot to the exit. You would like to fix the string so that the robot will reach the exit square. In one second, you can delete an arbitrary character, or add an arbitrary character at an arbitrary position. Find how quickly you can fix your friend's command string.

You do not care how long it takes the robot to find the exit, but only how long it takes to repair the command string.

Input

The first line of input contains the two integers n and m ($1 \leq n, m \leq 50$). Each of the next n lines contains m characters, describing the corresponding row of the grid. Empty cells are denoted as '`.`', the robot's initial position is denoted as '`R`', obstacles are denoted as '#', and the exit is denoted as '`E`'. The next and final line of input contains your friend's command string, consisting of between 1 andc50 characters, inclusive. It is guaranteed that the grid contains exactly one `R` and one `E`, and that there is always a path from `R` to `E`.

Output

Print, on a single line, a single integer indicating the minimum amount of time to fix the program.

Sample Input 1

```
3 3
R..
.#.
..E
LRDD
```

Sample Output 1

```
1
```

Sample Input 2

```
2 4
R.#.
#..E
RRUUDDRRUUUU
```

Sample Output 2

```
0
```

This page is intentionally left (almost) blank.

Problem D
Enclosure

In the Dark Forest, the territory you control is defined by the smallest convex polygon that contains all trees you control. Your power is defined by the area of the territory you control.

You currently control k out of n trees in the Dark Forest. What is the highest power you can achieve by gaining control over a single additional tree somewhere in the forest?

Input

The first line of input consists of two space-separated integers n and k ($3 \leq k < n \leq 100,000$). Next follow n lines each with two space-separated integers x_i and y_i ($|x_i|, |y_i| \leq 10^9$) specifying the locations of the n trees. You control the first k trees given in the list; the other $n - k$ trees do not belong to you. (Note that some of these may still be inside your territory.) It is guaranteed that no three trees have collinear locations.

Output

Print, on a single line, the maximum power you can achieve by gaining control over a single additional tree. The output should be rounded and displayed to exactly one decimal place.

Sample Input 1

```
5 3
-5 -5
-5 5
5 -5
-4 6
5 5
```

Sample Output 1

```
100.0
```

Sample Input 2

```
4 3
999999999 999999999
999999998 -1000000000
-999999999 999999998
-1000000000 -999999999
```

Sample Output 2

```
3999999992000000004.0
```

This page is intentionally left (almost) blank.

Problem E
Assigning Workstations

Penelope is part of the admin team of the newly built supercomputer. Her job is to assign workstations to the researchers who come here to run their computations at the supercomputer.

Penelope is very lazy and hates unlocking machines for the arriving researchers. She can unlock the machines remotely from her desk, but does not feel that this menial task matches her qualifications. Should she decide to ignore the security guidelines she could simply ask the researchers not to lock their workstations when they leave, and then assign new researchers to workstations that are not used any more but that are still unlocked. That way, she only needs to unlock each workstation for the first researcher using it, which would be a huge improvement for Penelope.

Picture by NASA via WikiMedia Commons

Unfortunately, unused workstations lock themselves automatically if they are unused for more than m minutes. After a workstation has locked itself, Penelope has to unlock it again for the next researcher using it. Given the exact schedule of arriving and leaving researchers, can you tell Penelope how many unlockings she may save by asking the researchers not to lock their workstations when they leave and assigning arriving researchers to workstations in an optimal way? You may assume that there are always enough workstations available.

Input

The input consists of:

- one line with two integers n ($1 \leq n \leq 300\,000$), the number of researchers, and m ($1 \leq m \leq 10^8$), the number of minutes of inactivity after which a workstation locks itself;
- n lines each with two integers a and s ($1 \leq a, s \leq 10^8$), representing a researcher that arrives after a minutes and stays for exactly s minutes.

Output

Output the maximum number of unlockings Penelope may save herself.

Sample Input 1

```
3 5
1 5
6 3
14 6
```

Sample Output 1

```
2
```

Sample Input 2

```
5 10
2 6
1 2
17 7
3 9
15 6
```

Sample Output 2

```
3
```

This page is intentionally left (almost) blank.

Problem F
Cameras

Your street has n houses, conveniently numbered from 1 to n. Out of these n houses, k of them have security cameras installed. Mindful of gaps in coverage, the Neighborhood Watch would like to ensure that every set of r consecutive houses has at least two different houses with cameras.

What is the minimum number of additional cameras necessary to achieve this?

Input

The first line of input contains three integers, n ($2 \leq n \leq 100,000$), k ($0 \leq k \leq n$), and r ($2 \leq r \leq n$). The next k lines of input contain the distinct locations of the existing cameras.

Output

Print, on a single line, a single integer indicating the minimum number of cameras that need to be added.

Sample Input 1

```
15 5 4
2
5
7
10
13
```

Sample Output 1

```
3
```

This page is intentionally left (almost) blank.

Problem G
Finbonacci Words

The Fibonacci word sequence of bit strings is defined as:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n \geq 2 \end{cases}$$

Here, $+$ denotes concatenation of strings. The first few elements are:

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	101101011011010110101
8	1011010110110101101011011010110110
9	1011010110110101101011011010110110101101011011010110101

Given a bit pattern p and a number n, how often does p occur in $F(n)$?

Input

The first line of each test case contains the integer n ($0 \leq n \leq 100$). The second line contains the bit pattern p. The pattern p is nonempty and has a length of at most $100,000$ characters.

Output

For each test case, display the number of occurrences of the bit pattern p in $F(n)$. Occurrences may overlap. The number of occurrences will be less than 2^{63}.

Sample Input 1

```
6
10
```

Sample Output 1

```
5
```

Sample Input 2

```
7
10
```

Sample Output 2

```
8
```

Sample Input 3

```
6
01
```

Sample Output 3

```
4
```

Sample Input 4

```
6
101
```

Sample Output 4

```
4
```

Sample Input 5

```
96
10110101101101
```

Sample Output 5

```
7540113804746346428
```

This page is intentionally left (almost) blank.

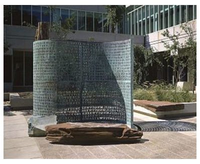


Figure H.1: The Kryptos statue, Langley, VA. Source Wikipedia

Problem H
NVWLS

NVWLS, or "No Vowels" puzzles are popular among puzzle enthusiasts. For example, consider the following no-vowels message:

```
BTWNSBTLSHDNGNDTHBSNCFLGHTLSTHNNCFQLSN
```

which is inscribed on the famous "Kryptos" statue located at the CIA's headquarters in Virginia. This message is derived from the following sentence by removing all vowels and spaces:

```
BETWEENSUBTLESHADINGANDTHEABSENCEOFLIGHTLIESTHENUANCEOFIQLUSION
```

Given a dictionary (a set of words that can be used to construct a sentence) and a message (which comes from a sentence which uses only those words, but with all vowels and spaces removed), reconstruct the original sentence using the dictionary words!

Input

The first line contains an integer n denoting the number of words in the dictionary. The next n lines each contain a dictionary word using one or more uppercase English letters. Each word contains at least one consonant.

For the purposes of this problem, the letters `A`, `E`, `I`, `O`, and `U` are vowels and all other letters are consonants.

The dictionary is followed by a single non-empty line of uppercase consonants representing the no-vowels message. It is guaranteed that the no-vowels message can be constructed in at least one way using only the dictionary words.

The total number of letters in all dictionary words is no greater than $100,000$. The total number of letters in the no-vowels message does not exceed $300,000$.

Output

Output a whitespace-separated sequence of dictionary words that yields the original no-vowels message when all spaces and vowels are removed. If there are multiple reconstructions, choose the one with the largest overall number of vowels. If there are still multiple reconstructions, you may output any one of them. No judge input will require your program to output more than $15,000,000$ characters.

Sample Input 1

```
11
BETWEEN
SUBTLE
SHADING
AND
THE
ABSENCE
OF
LIGHT
LIES
NUANCE
IQLUSION
BTWNSBTLSHDNGNDTHBSNCFLGHTLSTHNNCFQLSN
```

Sample Output 1

```
BETWEEN SUBTLE SHADING AND THE ABSENCE OF LIGHT LIES THE NUANCE OF IQLUSION
```

Sample Input 2

```
4
NA
NNANNA
NANNA
BATMAN
NNNNNNNNNNNNNNBTMN
```

Sample Output 2

```
NA NA NA NA NA NA NA NA NA NA NA NA NA BATMAN
```

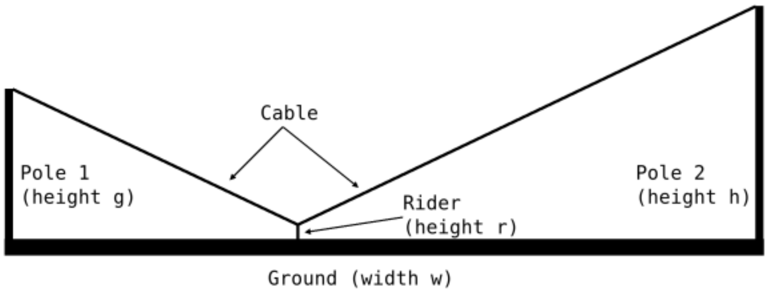


Figure I.1: A zipline, annotated with the four variables used to describe it.

Problem I
Zipline

A zipline is a very fun and fast method of travel. It uses a very strong steel cable, connected to two poles. A rider (which could be a person or some cargo) attaches to a pulley which travels on the cable. Starting from a high point on the cable, gravity pulls the rider along the cable.

Your friend has started a company which designs and installs ziplines, both for fun and for utility. However, there's one key problem: determining how long the cable should be between the two connection points. The cable must be long enough to reach between the two poles, but short enough that the rider is guaranteed to stay a safe distance above the ground. Help your friend determine these bounds on the length.

The cable connects to two vertical poles that are w meters apart, at heights g and h meters, respectively. You may assume that the cable is inelastic and has negligible weight compared to the rider, so that there is no sag or slack in the cable. That is, at all times the cable forms two straight line segments connecting the rider to the two poles, with the sum of the segments lengths equal to the total length of the cable. The lowest part of the rider hangs r meters below the cable; therefore the cable must stay at least r meters above the ground at all times during the ride. The ground is flat between the two poles. Please refer to the diagram in Figure I.1 for more information.

Input

The input starts with a line containing an integer n, where $1 \le n \le 1{,}000$. The next n lines each describe a zipline configuration with four integers: w, g, h, and r. These correspond to the variables described above. The limits on their values are: $1 \le w, g, h \le 1{,}000{,}000$, and $1 \le r \le \min(g, h)$.

Output

For each zipline, print a line of output with two lengths (in meters): the minimum and maximum length the cable can be while obeying the above constraints. Both lengths should have an absolute error of at most 10^{-6}.

Sample Input 1

```
2
1000 100 100 20
100 20 30 2
```

Sample Output 1

```
1000.00000000 1012.71911209
100.49875621 110.07270325
```

This page is intentionally left (almost) blank.

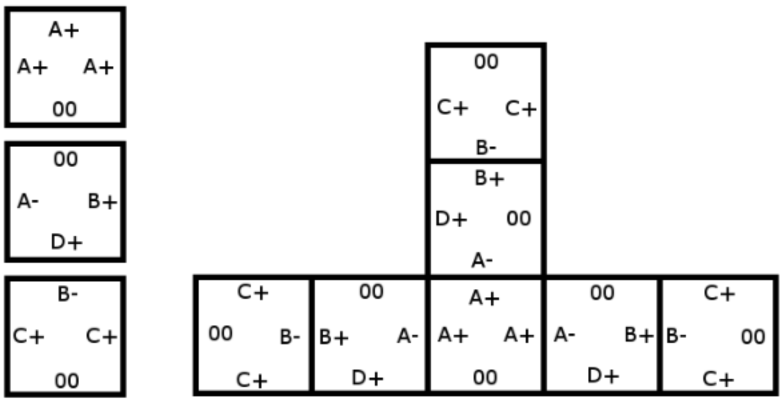


Figure J.1: Illustration of Sample Input 1.

Problem J
Self-Assembly

Automatic Chemical Manufacturing is experimenting with a process called self-assembly. In this process, molecules with natural affinity for each other are mixed together in a solution and allowed to spontaneously assemble themselves into larger structures. But there is one problem: sometimes molecules assemble themselves into a structure of unbounded size, which gums up the machinery.

You must write a program to decide whether a given collection of molecules can be assembled into a structure of unbounded size. You should make two simplifying assumptions:

1. the problem is restricted to two dimensions, and
2. each molecule in the collection is represented as a square.

The four edges of the square represent the surfaces on which the molecule can connect to other compatible molecules. In each test case, you will be given a set of molecule descriptions. Each type of molecule is described by four two-character connector labels that indicate how its edges can connect to the edges of other molecules. There are two types of connector labels:

- An uppercase letter $(A, \ldots, Z)$ followed by $+$ or $-$. Two edges are compatible if their labels have the same letter but different signs. For example, $A+$ is compatible with $A-$ but is not compatible with $A+$ or $B-$.
- Two zero digits 00. An edge with this label is not compatible with any edge (not even with another edge labeled 00).

Assume there is an unlimited supply of molecules of each type, which may be rotated and reflected. As the molecules assemble themselves into larger structures, the edges of two molecules may be adjacent to each other only if they are compatible. It is permitted for an edge, regardless of its connector label, to be connected to nothing (no adjacent molecule on that edge).

Figure J.1 shows an example of three molecule types and a structure of bounded size that can be assembled from them (other bounded structures are also possible with this set of molecules).

Input

The input consists of a single test case. A test case consists of two lines. The first contains an integer n, $(1 \leq n \leq 40,000)$ indicating the number of molecule types. The second line contains n eight-character strings, each describing a single type of molecule, separated by single spaces. Each string consists of four two-character connector labels representing the four edges of the molecule in clockwise order.

Output

Display the word `unbounded` if the set of molecule types can generate a structure of unbounded size. Otherwise, display the word `bounded`.

Sample Input 1

```
3
A+00A+A+ 00B+D+A- B-C+00C+
```

Sample Output 1

```
bounded
```

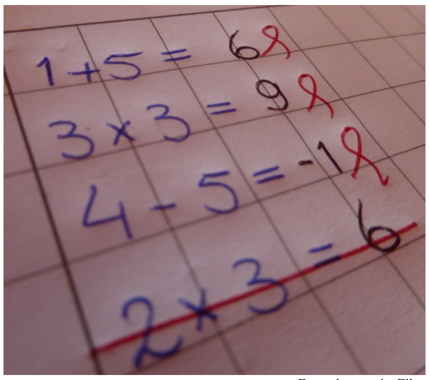
Sample Input 2

```
1
K+K-Q+Q-
```

Sample Output 2

```
unbounded
```

Problem K
Elementary Math



Example exam by Ellen

Ellen is teaching elementary math to her students and the time for the final exam has come. The exam consists of n questions. In each question the students have to add ($+$), subtract ($-$) or multiply ($*$) a pair of numbers.

Ellen has already chosen the n pairs of numbers. All that remains is to decide for each pair which of the three possible operations the students should perform. To avoid students getting bored, Ellen wants to make sure that the n correct answers to her exam are all different.

Please help Ellen finish constructing the exam by automating this task.

Input

The input consists of:

- one line with one integer n ($1 \leq n \leq 2\,500$), the number of pairs of numbers;
- n lines each with two integers a and b ($-10^6 \leq a, b \leq 10^6$), a pair of numbers used.

Output

For each pair of numbers (a, b) in the same order as in the input, output a line containing a valid equation. Each equation should consist of five parts: a, one of the three operators, b, an equals sign ($=$), and the result of the expression. All the n expression results must be different.

If there are multiple valid answers you may output any of them. If there is no valid answer, output a single line with the string "`impossible`" instead.

Sample Input 1	Sample Output 1

4
1 5
3 3
4 5
-1 -6
	1 + 5 = 6
	3 * 3 = 9
	4 - 5 = -1
	-1 - -6 = 5

Sample Input 2	Sample Output 2

4
-4 2
-4 2
-4 2
-4 2
	impossible

This page is intentionally left (almost) blank.