Freshmen Programming Contest 2024

FPC 2024

Problems

A Annoying Alliterations

Time limit: 2s

Bajtazar is a member of the jury for the Find Your Palace Contest, a prestigious programming competition that offers a luxurious palace to the winner. He is now trying to come up with a name for his problem. According to Bajtazar, a good problem name consists of exactly two words. Additionally, Bajtazar finds alliterations very annoying, so he will remove the first letter from both words until the first letter of the two words is different or one of them becomes empty. After this operation, Bajtazar defines the *goodness* of the problem name as the sum of lengths of the two words.

The prizewinning participant of Bajtazar's contest will win this colossal castle.
CC BY 2.0 by Fred Romero on Flickr

He has prepared a list of n words, and started to wonder what is the maximum goodness that can be achieved using the words from the list. Bajtazar himself does not have time to answer that question as he is busy reinforcing the tests with nasty edge cases. Help Bajtazar by finding the maximum goodness that can be achieved.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of words.
- n lines, each with a word w ($1 \leq |w| \leq 10^6$).
  Each word only consists of English lowercase letters (`a`–`z`).

The total number of characters in the n words is at most 10^6.

Output

Output the maximum goodness that can be achieved.

Sample Input 1	Sample Output 1
3 amsterdam is amazing	12

Sample Input 2	Sample Output 2
3 bbb abbba aaba	8

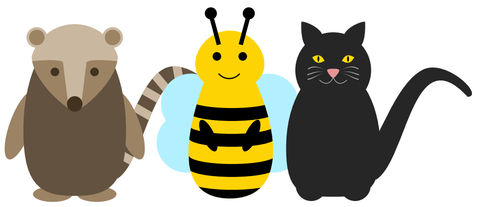
Sample Input 3	**Sample Output 3**

`2`
`abcdefghijklmnopqrstuvwxyz`
`abcdefghijklmnopqrstuvwxyz`
	`0`

B Building Pyramids

Time limit: 1s

At long last, the cats have finally set aside their squabbles with the coatis and the bees. They decide to build a monument to commemorate the end of their feud. To show that such different animals can live together in harmony, the Great Architect Principal Coati decides that this monument should consist of two completely different three-dimensional shapes: the triangular pyramid (for having very sharp corners) and the sphere (for having no corners at all). The idea is to create a large regular tetrahedron out of many small spheres, similar to the one shown in Figure B.1.


Alicia, Beesworth, and Darabella.
TikZlings LPPL-1.3c by samcarter on GitHub

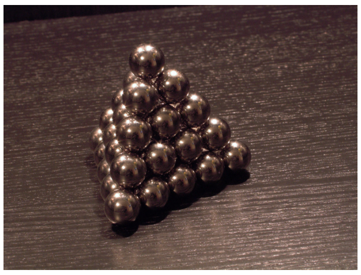


Figure B.1: A triangular pyramid of spheres, corresponding to the second sample input. One edge of the pyramid contains 5 spheres, so the entire pyramid consists of 35 spheres.
Image CC BY-NC 2.0 by Amafirlian on Flickr

Alicia the coati, Beesworth the bee, and Darabella the cat would like this monument to be as large as possible. They have designated a piece of land to place the monument on, and have determined how many spheres should be in one of the edges based on where the monument will be placed. How many spheres will the entire pyramid contain, knowing the number of spheres in one of the edges?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^6$), the number of spheres in one of the edges of the pyramid.

Output

Output the number of spheres in the entire pyramid.

Sample Input 1	Sample Output 1
3	10

Sample Input 2

```
5
```

Sample Output 2

```
35
```

Sample Input 3

```
7777
```

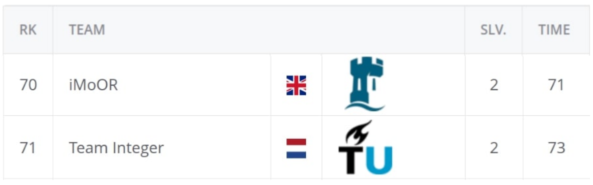
Sample Output 3

```
78424644529
```

C Curious Jury

Time limit: 3s

There are multiple ways to break ties based on submit time in competitive programming contests. In particular, the time penalty can either be the sum of all the times that you submit a correct solution (like in this contest!), or it can be the time that the last accepted submission of a team came in.



RK	TEAM			SLV.	TIME
70	iMoOR			2	71
71	Team Integer			2	73

Almost fixed points in NWERC 2015.
Compiled from the NWERC 2015 Scoreboard

As a jury member of a prestigious team contest, you are really excited by "fixed points". These are places in the scoreboard where the *rank* of a particular team is equal to their penalty time. Because all teams solved all problems, the rank of a team is equal to the number of teams who had a strictly lower penalty time + 1.

For each team, you know the sum of submit times of submissions s, and a last submit time l. The contest consists of more than one problem, so it must hold that $l < s$ for each team. Additionally, we assume that the sum of submit times does not exceed the number of teams.

For example, say the penalty times for teams A, B, and C are 1 minute, 2 minutes, and 1 minute, respectively. Teams A and C both share rank 1, because no other teams have a strictly lower penalty time. They both also have penalty time 1, so they both form a fixed point on the scoreboard. There are two other teams which have a strictly lower penalty time than team B, so team B has rank 3, and its penalty time is 2, so it does not constitute a fixed point. In this example, there are 2 fixed points.

For each team, you decide to arbitrarily pick between the two ways of calculating the penalty time, to make fixed points.

For a given way of picking the sum or last submit time for each team, define g as the number of fixed points the scoreboard has.

Find the sum of g over all ways of choosing penalty times.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of participating teams.
- n lines with two integers l and s ($1 \leq l < s \leq n$), the last submit time and the sum of submit times for a team.

Output

Output the sum of g, the number of fixed points, over all 2^n ways of choosing penalty times.

Because the answer can be huge, output the answer modulo the prime $10^9 + 7$.

Sample Input 1

```
3
1 2
2 3
1 3
```

Sample Output 1

```
16
```

Sample Input 2

```
10
1 6
2 7
3 8
1 2
5 9
9 10
1 10
2 8
3 4
4 5
```

Sample Output 2

```
4752
```

D Dragged-out Duel

Time limit: 1s

Your friend Guile and you have decided to spend an afternoon playing the new Street Fighter game, but you both want to start on the left, because you have both only practiced all your combos starting from the left. You decide to determine who starts on the left like true men – with an intense grueling, and exhausting duel of rock-paper-scissors.

Screenshot taken from the animated television series "Arnold", developed by Kathy Waugh and produced by WGBH, modified using imgflip meme generator

The duel will be a best-of-n, consisting of n single rounds of rock-paper-scissors (yes, you may have to play rock-paper-scissors *ten thousand times*, truly a war of attrition), and whoever wins more rounds, gets to start on the left. If you both pick the same option in an individual game, it is not replayed. To keep track of who wins, you decide to write a program, that determines whether you beat Guile in this great duel.

As a reminder: rock beats scissors, scissors beats paper, paper beats rock. If you both make the same choice, the round is a draw.

Input

The input consists of:

- One line with a single integer n ($1 \leq n \leq 10\,000$), the number of individual games.
- Two lines with n characters, each character being either '`R`' for rock, '`P`' for paper, or '`S`' for scissors. The first line represents your choices in each round, and the second line represents Guile's choices in each round.

Output

If you have won the most rounds of rock-paper-scissors, output "`victory`".
If Guile has won the most rounds of rock-paper-scissors, output "`defeat`".

It is guaranteed that the both of you will not have the same number of wins.

Sample Input 1

```
5
RRSSP
SSSSS
```

Sample Output 1

```
victory
```

Sample Input 2

```
6
PRSRPS
PSSPPR
```

Sample Output 2

```
defeat
```

This page is intentionally left blank.

E European Election

Time limit: 5s

The European Parliament in Strasbourg. CC BY 3.0 by Ralf Roletschek on Wikimedia

The European presidential election of 2042 are coming up soon. The First Presidential Committee has devised the most objective and representative election method. Every candidate is individually compared to every other candidate. The winner of the election is going to be the one candidate who is preferred over all other candidates.

To facilitate this, on their ballot, each voter lists candidates in order of preference. From those rankings, the head-to-head preferences of each voter are inferred. Not all candidates have to be ranked by every voter – omitted candidates are considered to all be ranked last by the voter (with no preference between them). A candidate is better than another candidate if more people prefer that candidate over the other one. If a candidate is better than every other candidate, they win the election.

Consider the second sample input, where there are seven voters and three candidates. In the head-to-head battle between candidates 1 and 2, four voters prefer candidate 1 and two voters prefer candidate 2 (one voter is indifferent about both), so candidate 1 wins that head-to-head match-up. Between candidates 1 and 3, four people prefer candidate 1, and three people prefer candidate 3, thus candidate 1 is better. Therefore, candidate 1 wins the election.

In case no candidate wins against all other candidates, then the results of the election are inconclusive – there is no winner. If, in a head-to-head battle, two candidates have the same number of preferences, then neither of them is considered better than the other. Thus, neither of them would be able to win the election.

Given the contents of each ballot, determine whether the election has a conclusive winner, and if so, who that winner is.

Input

The input consists of:

- One line with integers n and k ($1 \leq n \leq 2000$, $1 \leq k \leq 200$), denoting the number of ballots cast and the number of candidates.
- n lines, one for each ballot.

  Each line starts with an integer v ($0 \leq v \leq 200$), the number of preferences on this ballot.

  The remainder of this line has v integers p ($1 \leq p \leq k$), denoting the preferences of this voter – the first integer is the number of the voter's first preference candidate, the second integer is their second preference, and so on.

  All the candidates that are unranked (i.e. not mentioned in the ballot) are considered to be all ranked last (i.e. tied with each other).

Output

If the election has a conclusive winner, output the number of the candidate who wins the election. Otherwise, output "`impossible`".

Sample Input 1

```
6 4
2 1 2
4 2 4 3 1
3 3 4 2
2 4 1
1 4
3 4 3 2
```

Sample Output 1

```
4
```

Sample Input 2

```
7 3
3 1 2 3
2 2 1
3 3 1 2
3 1 3 2
2 3 2
1 3
2 1 3
```

Sample Output 2

```
1
```

Sample Input 3

```
3 3
3 1 2 3
3 2 3 1
3 3 1 2
```

Sample Output 3

```
impossible
```

Sample Input 4

```
4 3
3 1 2 3
2 1 2
2 2 1
3 2 1 3
```

Sample Output 4

```
impossible
```

F Flag Rotation

Time limit: 2s

A common pattern for country flags is three stripes. Some flags are really similar, and have exactly the same stripes in the same order, but the flag is rotated by 90 degrees. Using a sequence of colours, we can make a flag on a square grid, covering each row with the colours in the sequence. Another way to make a similar flag (that is rotated by 90 degrees), is by covering each column with the colours in the sequence.

In preparation for the Flags You Paint for Countries meeting, you notice a mistake for one of the flags: it should have been rotated counter-clockwise by 90 degrees! The square canvas is already stuck to a wall using super glue, so you can not simply take it off and rotate it. Therefore, you want to know how much work it would be to repaint one flag to the other. In one second, you can repaint one of the 1×1 squares to any colour. What is the minimum number of seconds you need to repaint the flag? As an example, for the flag in the first sample input (shown in Figure F.1), 34 squares need to be repainted.

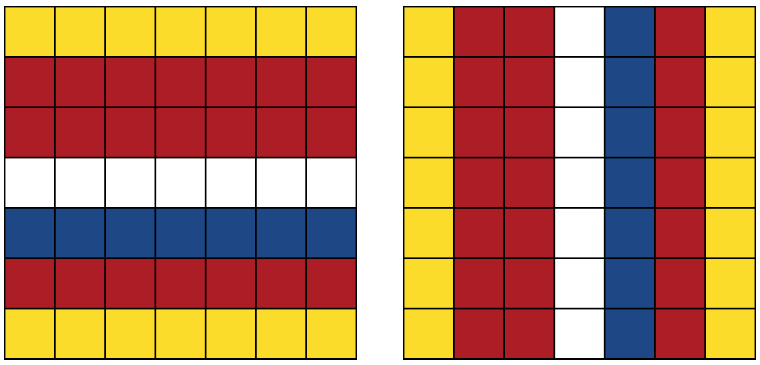


Figure F.1: Two flags corresponding to the first sample input. Colour 1 corresponds to yellow, 2 to red, 3 to white, and 1000 to blue. When the first flag should be rotated counter-clockwise by 90 degrees without actually rotating the canvas, 34 squares need to be repainted.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of stripes in the flag.
- One line with n integers a ($1 \leq a \leq 10^9$), the colours of the stripes.

Output

Output the number of cells that need to be repainted when the flag should be rotated counter-clockwise by 90 degrees.

Sample Input 1	Sample Output 1
7 1 2 2 3 1000 2 1	34

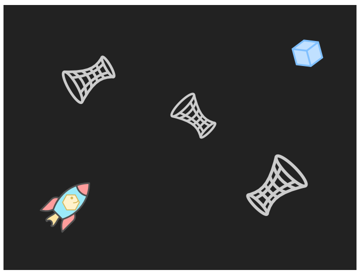
Sample Input 2	**Sample Output 2**
4 10 12 12 10	8

Sample Input 3	**Sample Output 3**
5 6 5 5 6 5	12

G Galactic Expedition

Time limit: 8s

The inhabitants of Flatland have finally managed to invent spacecraft, and send a hexagon to embark on a brave journey to explore their two-dimensional outer space. Their goal is to reach an ancient relic that might teach them about the absurd idea of a "third dimension". You are head of operations at the home base, coordinating the movements of the spaceship.


A happy hexagon going on an expedition to find the ancient relic. Rocket modified from Optima GFX on IconFinder

The spaceship has enough fuel to travel a certain distance in space, but it can be refuelled at the home base if necessary. Chances are high that the spaceship does not have enough fuel to travel to the relic directly, but you quickly discover that outer space is scattered with wormholes that allow the spaceship to teleport from one location to another. Your team manages to draw out a map with all the warp-points they can observe, but it is impossible to figure out which pairs of warp-points are connected to each other, except by flying through them. Time to start exploring!

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads from the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends:

- One line with two integers n and d ($2 \leq n \leq 500$, n is even, $2 \leq d \leq 10^6$), the number of points in outer space and the distance that your spaceship can travel with a full fuel tank.
- n lines, the ith of which contains two integers x and y ($|x|, |y| \leq 10^6$), the coordinates of the ith point in outer space. Point 1 corresponds to the home base, the relic is located at point n, and all other points are warp-points.

Then, your program should start travelling between these points, starting from point 1. By printing a line containing an integer i ($1 \leq i \leq n$), your spaceship will travel to the ith point.

When you travel to a warp-point, you will teleport through it and the interactor will respond with an integer j ($1 < j < n$, $i \neq j$), indicating the warp-point that you have teleported to. This teleportation does not consume any fuel. Travelling to warp-point j later will teleport you back to warp-point i. Teleportation only occurs when you explicitly travel to a warp-point, not when you happen to fly over other warp-points.

When you travel to point 1, your fuel tank will be refilled when you reach it, and the interactor will always respond with 1.

When you travel to point n, the interaction will stop.

It is guaranteed the shortest travel distance from point 1 to any other point is at most $\frac{d}{2}$ when using the wormholes optimally.

The interactor is not adaptive: the wormhole links are fixed up front, and do not depend on how you decide to travel.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Running out of fuel or travelling to point 1 more than $\frac{n}{2}$ times will result in a wrong answer.

Sample Interaction 1

Read	Write

6 30
0 0
0 5
100 0
105 0
0 200
0 205	

	2
3	
	4
5	
	6

Sample Interaction 2

Read	Write

6 20
0 0
0 5
95 100
-5 0
-100 0
100 100	

	4
5	
	5
4	
	1
1	
	2
3	
	6

H Horrendous Mistake

Time limit: 3s

While looking through the Grand Archive of Problematic Code, you found this horrendous mistake in one of the code snippets. In this code, they tried to calculate the sum of an array, but got indices and values mixed up. The code of this `sum` function is shown in Figure H.1.

```cpp
long long sum(vector<int> a) {
    long long ans = 0;
    for (int x : a)
        ans += a[x];
    return ans;
}
```

```python
def sum(a: list[int]) -> int:
    ans = 0
    for x in a:
        ans += a[x]
    return ans
```

```java
long sum(int[] a) {
    long ans = 0;
    for (int x : a)
        ans += a[x];
    return ans;
}
```

```kotlin
fun sum(a: List<Int>): Long {
    var ans = 0L
    for (x in a)
        ans += a[x]
    return ans
}
```

Figure H.1: The function `sum` intends to calculate the sum of an array, but `x` refers to a value in the array, instead of an index. The code is shown in C++ and Python in the top row, and Java and Kotlin in the bottom row.

You wonder how this function behaves exactly, and decide to thoroughly test it. Starting with some initial array, you apply a sequence of updates. For each update, you change one of the values of the array. You wonder what the value of the function is after each update.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the length of the array.
- One line with n integers a ($0 \leq a < n$), the values in the initial array.
- One line with an integer t ($1 \leq t \leq 2 \cdot 10^5$), the number of updates in your sequence of testing.
- t lines, each with two integers x and v ($0 \leq x, v < n$), indicating that the xth entry in the array is updated to the new value v.

Note that the array uses 0-based indexing.

Output

For each update, output the return value of the function `sum` applied to the array after applying the update.

Sample Input 1

```
5
0 0 3 2 0
4
1 1
0 4
3 4
1 4
```

Sample Output 1

```
6
10
9
8
```

Sample Input 2

```
5
4 2 2 4 2
3
0 1
4 3
2 3
```

Sample Output 2

```
10
13
16
```

I Intelligence Exploration

Time limit: 2s

John is a researcher working in the field of Artificial Intelligence. At the moment he needs to analyze reasoning abilities of a new Limitless Logic Machine, he wants to check how well it can understand when one thing implies another.

A picture of John. Generated using Microsoft Copilot Designer

The scientist is planning to use a long array a consisting of zeroes and ones for his experiment. During the experiment, he will make multiple queries to the machine, where each query is defined by a pair l, r. For each query, the machine will be asked to compute the value of the implication of the subarray $a_l \to a_{l+1} \to \ldots \to a_r$. John asks you, as his AI Application Project Junior Engineer, to compute the correct answers for these queries, so he can use them to validate his machine.

The implication operator is defined as follows:

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

In this problem, we assume a left-to-right evaluation order, so the order of the operations is $((\ldots((a_l \to a_{l+1}) \to a_{l+2}) \to \ldots) \to a_r)$.

Input

The input consists of:

- One line with an integer n ($2 \le n \le 10^5$), the size of the array.
- One line with n integers a ($a \in \{0, 1\}$), the values in the array.
- One line with an integer q ($1 \le q \le 10^5$), the number of queries.
- q lines, each with two integers l and r ($1 \le l < r \le n$), describing a query.

Output

For each query, output the result of the corresponding implication.

Sample Input 1

```
5
0 0 1 1 0
5
1 2
2 3
4 5
3 4
1 5
```

Sample Output 1

```
1
1
0
1
0
```

Sample Input 2

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

Sample Output 2

```
1
1
0
0
1
```

J Jailbreak

Time limit: 2s

CC BY-SA 2.0 by seabamirum on Flickr

You are one of the greatest computer scientists of the 21st century, and just discovered a polynomial-time algorithm for integer factorization. Unfortunately, the secret service noticed your work. They confiscated the algorithm and put you in an underground jail, the Federal Prison for Criminals. You are determined to escape, publish your findings,[1] and seek justice.

The only exits in the jail through which you can escape are in the top storey of the jail, while you currently are in the leftmost cell of the bottom storey in the jail, h storeys below the ground. To climb up through holes in the ceiling, you need a ladder. However, since you have practiced breaking a fall, and there are no two holes directly below each other, you can jump down through a hole without a ladder. Given the layout of the jail, determine whether it is possible to escape the jail.

The ladders can be found in the jail. You cannot carry another ladder up or down through a hole, or retrieve a ladder from a different storey, but you can carry them to different cells on the same (wall-enclosed) part of the same storey. You can jump over (arbitrarily many) holes in the floor/ceiling.

As an example, consider the first sample input. You can use the ladder next to you to go up one storey. Then you cannot go up again: although there is a hole, you have no ladder. But you can go down and then up twice, and finally escape.

Input

The input consists of:

- One line with two integers w and h ($3 \leq w \leq 10^5$, $1 \leq h \leq 10^5$, $w \cdot h \leq 3 \cdot 10^5$), the width and height of the jail.
- $2h + 1$ lines with w characters:
  - On odd-numbered lines, the characters are either '`-`' or '`.`', representing the ceiling or a hole in the ceiling, respectively.
    The last line will only contain '`-`', to represent the floor.
  - On even-numbered lines, the characters are either '`|`', '`.`', or '`L`', representing a wall, an empty space, or a ladder, respectively.
    The first and last character will always be a wall.

It is guaranteed that there is not a wall in the cell in a storey (even-numbered line) directly above and below a hole in a ceiling (odd-numbered line).
It is guaranteed that there are no two holes in the ceilings directly below each other.
It is guaranteed that the starting cell (bottom left cell) is not a wall.

[1]No worries: while the algorithm runs in polynomial time, it is not fast in practice.

Output

If it is possible to escape the jail, output "`possible`". Otherwise, output "`impossible`".

Sample Input 1

```
10 3
-----.----
|.|L....||
----.--.--
|.....|L.|
---.-.--.-
|.L.|..L.|
----------
```

Sample Output 1

```
possible
```

Sample Input 2

```
7 3
---.---
|..L|.|
--.----
|.L..||
-.-.---
|.|L..|
-------
```

Sample Output 2

```
impossible
```

K Kangaroo Race

Time limit: 5s

An Austrian kangeroo.
Map CC BY-SA 3.0 by TUBS on Wikimedia Commons, kangaroo by brgfx on Freepik

The country of Austria is well known for their kangaroo population. In order to stay in good shape, the kangaroos each have an athletics track to practice for the Annual Austrian Pogostick Jumping Event.

Each athletics track consists of n segments, each 1 meter in length. These segments are numbered from 1 to n, in order. The track is cyclic, so after segment n follows segment 1 again.

On each track, a kangaroo is located in one of the segments. The kangaroo can make some finite number of jumps. In each jump, if the kangaroo is currently in segment y, it will jump $y(y-1)$ segments ahead. Your task is to determine the minimum number of jumps needed for the kangaroo to reach the segment numbered 1.

Since the kangaroo population in Austria is quite large, you are asked to solve this problem for many different kangaroos on different athletics tracks.

Input

The input consists of:

- One line with an integer k ($1 \leq k \leq 10^5$), the number of kangaroos.
- k lines with two integers n and x ($1 \leq x \leq n \leq 10^{18}$), the number of segments in one of the athletics tracks and the kangaroo's initial position on this track.

Output

For each kangaroo, if the kangaroo can reach can reach the segment numbered 1 in a finite number of jumps, output the minimum number of jumps needed. Otherwise, output "`impossible`".

Sample Input 1	Sample Output 1

4
5 2
6 2
8 3
12345678910 1
	2
	impossible
	1
	0

Notes

The intermediate values of your calculation may become larger than what fits in 64-bit integers. To store these large intermediate values, use `__int128` in C++ or `java.math.BigInteger` in Java/Kotlin. In Python, integers have arbitrary size by default.