Student Name:

Student ID :

Problem Statement A1

The ground floor of an old building has recently been repurposed as a restaurant. The restaurant floor is a rectangle that is divided into R rows by C columns of unit squares. Some unit squares are walls ('#'), others contain lamps ('O'), and the rest is empty ('.'). You are given R, C, and the floor plan itself in the vector<string> roomPlan.

Each lamp has 2 LED focus lights, so that it illuminates its own square plus any other empty squares in its row and column - but only until the first wall in each direction.

Compute and return the number of empty squares that will remain in the dark, even if all lamps are turned on.

Definition

Class: DinnerLighting
Method: countDarkSquares
Parameters: int, int, vector <string >
Returns: int
Method signature: int countDarkSquares (int R, int C, vector<string> roomPlan)
(be sure your method is public)

Notes

- The character used to represent a lamp is 'O', the capital letter oh (and not '0', the digit zero).

Constraints

- R will be between 1 and 50, inclusive.
- C will be between 1 and 50, inclusive.
- roomPlan will contain R elements.
- Each element of roomPlan will contain C characters.
- Each character in roomPlan will be '#', 'O', or '.'.

Examples

0) 3
4
```
{"....",
 "....",
 "...."}
```
Returns: 12
There are no lights, the whole restaurant is in the dark.

1)
```
      4
      5
{".....",
 ".O...",
 ".....",
 "....."}
```
Returns: 12

A single light. When we turn it on, the restaurant looks like we show here, with '*' denoting cells that have light.

```
.*...
*O***
.*...
.*...
```

2)
```
      4
      5
{".....",
 ".OO..",
 ".....",
 "....."}
```
Returns: 9

The second light illuminates the three cells in that row, so we have 9 dark cells.

3)
```
      4
      5
{".....",
 ".O#..",
 ".#.O.",
 "....."}
```
In this case, the walls block some of the cells in the same row/column of a light.

Returns: 9

4)
```
      4
      5
{".....",
 ".O...",
 "...O.",
 "....."}
```
Returns: 6

5)
```
      1
      1
{"O"}
```
Returns: 0

Problem Statement A2

Let X and Y be two words of equal length, consisting of uppercase English letters only. The two words are called **related** if there is a permutation P of the English alphabet with the following property:

if each character c in X is replaced by the character P(c), we obtain Y.

In other words, X and Y are related iff the following holds: whenever two letters of X are equal, the corresponding two letters of Y must be equal, and vice versa. For example, consider the string "ABCA". We can choose to replace each 'A' by a 'F', each 'B' by a 'B', and each 'C' by a 'G', obtaining the string "FBGF". Thus the words "ABCA" and "FBGF" are related. The words "ABCA" and "EFGH" are not related, because the two 'A's in the first string must correspond to the same letter in the second string. The words "ABCA" and "EEEE" are not related, because different letters in the first string must correspond to different letters in the second string.

You are given two words A and B of the same length. These words only contain English letters from 'A' to 'I', inclusive. (That is, only the first 9 letters of the alphabet are used.)

You want to change A and B into two words that are related. The only allowed change is to choose some indices (possibly none) and to remove the characters at those indices from each of the words. (I.e., the removed characters must be at the same positions in both words. For example, it is not allowed to only remove character 0 of A and character 3 of B.) For example, if A="ABAC", B="AHHA" and the chosen indices are 0 and 2, then the resulting words will be "BC" and "HA". Our goal is to choose as few indices as possible, given that after the erasing we want to obtain two related words. Compute and return the smallest possible number of chosen indices.

Definition

Class:	RelatedWords
Method:	numRemovals
Parameters:	string, string
Returns:	int
Method signature:	int numRemovals(string A, string B)

(be sure your method is public)

Constraints

- A will contain between 1 and 50 characters, inclusive.
- A and B will be of the same length.
- A will contain characters from 'A' to 'I' only.
- B will contain characters from 'A' to 'I' only.

Examples

0) “DD"
"FF"

Returns: 0

The given words are related without any removals. Any mapping with 'D' mapped to 'F' changes A to B.

1) "AAAA"
"ABCD"

Returns: 3

We can choose any three indices.

2) "AAIAIA"
"BCDBEE"

Returns: 3

One possible triple of indices is (1, 2, 5) (0-based indices).

3) "ABACDCECDCDAAABBFBEHBDFDDHHD"
"GBGCDCECDCHAAIBBFHEBBDFHHHHE"

Returns: 9

Problem Statement B1

A rat is placed in a rectangular maze consisting of NxM squares. Each square either contains a wall or is empty. The maze is structured in such a way that for any two empty squares, there exists exactly one path between them. A path is a sequence of pairwise distinct empty squares such that every two consecutive squares are adjacent. Two squares are adjacent if they share a common edge.

One of the empty squares in the maze contains a piece of cheese. The rat's goal is to reach that square without visiting the same square twice. The rat can only move between adjacent squares. Since the rat is hungry and have been trained in the lab for a while, it is guaranteed to achieve her goal.

As the rat moves from his starting point to the cheese, she may encounter some squares where he must choose between several adjacent squares to continue. This happens when the rat steps into a square which has more than one adjacent empty square, excluding the square from which he came, or when she has more than one adjacent empty square at the start. These situations are called "decisions" and the rat <u>will always make the right choice.</u>

You are given a String[] maze representing the maze. It contains N elements, each containing M characters. Empty squares are denoted by '.', walls are denoted by uppercase 'X', the rat's starting point is denoted by 'R', and the square containing the cheese is denoted by '*'. Return the number of decisions the rat will make on her way to the cheese.

Definition

Class:	GetCheese
Method:	decisions
Parameters:	vector <string>
Returns:	int
Method signature:	int decisions(vector <string> maze)

(be sure your method is public)

Constraints

- maze will contain between 1 and 50 elements, inclusive.
- Each element of maze will contain between 1 and 50 characters, inclusive.
- Elements of maze will be of the same length.
- maze will contain only '.', 'X', 'R' or '*' characters.
- There will be exactly one '*' character in maze.
- There will be exactly one 'R' character in maze.
- For every pair of empty squares in the maze, there will exist exactly one path between them.

Examples

0) `{"*.R"}`

Returns: 0

From each square, the rat can only move to one other square, so it never has to make any decisions.

1)
```
{"*.R",
 ".X."}
```
Returns: 1
The rat has to make a decision right at the start.

2)
```
{"...",
 "XRX",
 "..*"}
```
Returns: 2
The rat makes decisions at both squares before reaching the cheese.

3)
```
{".X.X......X",
 ".X*.X.XXX.X",
 ".XX.X.XR...",
 "......XXXX."}
```
Returns: 3

4)
```
{"..........*",
 ".XXXXXXXXXX",
 "...........",
 "XXXXXXXXXX.",
 "R.........."}
```
Returns: 0

Problem Statement B2

You are the health and safety officer of a private primary school and it is your duty to organize evacuation drills in case of an emergency.

You want all teachers to be notified of an emergency event as quickly as possible, but you want for kids to be calm, so instead of using a loud evacuation alarm you organized in a tree-like structure to quickly but discretely report the event to all the teachers: each teacher has exactly one direct emergency leader, no teacher is his own direct or indirect leader, and every teacher is your direct or indirect subordinate.

In the event of an emergency, you will make a phone call to each of your direct subordinates, one at a time. After hearing the news, each subordinate must notify each of his direct subordinates, one at a time.

- each person will only call the teachers that they lead
- each phone call takes exactly one minute.

The process continues this way until all the teachers are aware of the emergency.

Note that there may be multiple phone calls taking place simultaneously. Return the minimum amount of time, in minutes, required for this process to be completed.

Teachers will be numbered starting from 1, while you will be numbered 0. Furthermore, every leader is numbered lower than his or her direct subordinates. You are given a vector <int> **leaders**, the *ith* element of which is the direct leader of teacher *i*. The first element of leaders will be -1, since you are the top of the tree.

Definition

Class:	EmergencyDrill
Method:	reportTime
Parameters:	vector <int>
Returns:	int

Method signature: int reportTime(vector <int> leaders)

(be sure your method is public)

Constraints

- leaders will contain between 1 and 50 elements, inclusive.
- Element i of leaders will be between 0 and i-1, inclusive, except for the first element which will be -1.

Examples

0) {-1, 0, 0}

Returns: 2

Call subordinate 1 at time 0 and then subordinate 2 at time 1. By time 2, both subordinates will be aware of the emergency.

1) {-1, 0, 0, 2, 2}
Returns: 3
This time call teacher 2 first, and then teacher 1 at time 1. After hearing the news, teacher 2 will call teacher 3 at time 1 and teacher 4 at time 2. It takes 3 minutes for everybody to know about the emergency.

2) {-1, 0, 1, 2, 3}
Returns: 4
Everyone in the school has only one subordinate, resulting in a chain of phone calls.

3) {-1, 0, 0, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 7, 7, 8, 12, 13, 14, 16, 16, 16}
Returns: 7