Happy Ladybugs

by nabila_ahmed

Happy Ladybugs is a board game having the following properties:

- The board is represented by a string, b, of length n. The i^{th} character of the string, b_i, denotes the i^{th} cell of the board.
  - If b_i is an underscore (i.e., `_`), it means the i^{th} cell of the board is empty.
  - If b_i is an uppercase English alphabetic letter (i.e., `A` through `Z`), it means the i^{th} cell contains a ladybug of color b_i.
  - String b will not contain any other characters.
- A ladybug is *happy* only when its left or right adjacent cell (i.e., $b_{i\pm1}$) is occupied by another ladybug having the same color.
- In a single move, you can move a ladybug from its current position to any empty cell.

Given the values of n and b for g games of Happy Ladybugs, determine if it's possible to make all the ladybugs happy. For each game, print `YES` on a new line if all the ladybugs can be made happy through some number of moves; otherwise, print `NO` to indicate that no number of moves will result in all the ladybugs being happy.

Input Format

The first line contains an integer, g, denoting the number of games. The $2 \cdot g$ subsequent lines describes a Happy Ladybugs game in the following format:

1. The first line contains an integer, n, denoting the number of cells on the board.
2. The second line contains a string, b, describing the n cells of the board.

Constraints

- $1 \leq g \leq 100$
- $1 \leq n \leq 100$
- It is guaranteed that string b consists of underscores and uppercase English alphabetic letters (i.e., `_` and `A` through `Z`).

Output Format

For each game, print `YES` on a new line if it is possible to make all the ladybugs *happy*; otherwise, print `NO`.

Sample Input

```
4
7
RBY_YBR
6
X_Y__X
2
__
6
B_RRBR
```

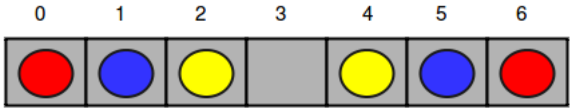
Sample Output

```
YES
NO
YES
YES
```

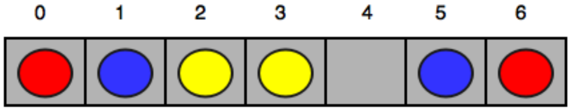
Explanation

The first three games of Happy Ladybugs are explained below:

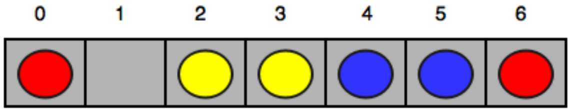
1. Initial board:



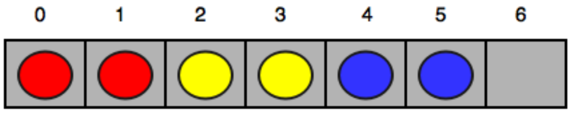
After the first move:



After the second move:



After the third move:



Now all the ladybugs are happy, so we print `YES` on a new line.

2. There is no way to make the ladybug having color `Y` happy, so we print `NO` on a new line.
3. There are no unhappy ladybugs, so we print `YES` on a new line.

Contest ends in 4 days

Submissions: 3780

Max Score: 27

Difficulty: Easy

Rate This Challenge:

☆☆☆☆☆

More

Current Buffer (saved locally, editable)

Java 7

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
import java.util.ArrayList;

public class Solution {

    public static void main(String[] args) {
        int Color[27]; // this array would contain number of occurances for the color.

        Scanner in = new Scanner(System.in);

        int Q = in.nextInt();
        for(int a0 = 0; a0 < Q; a0++){
            int n = in.nextInt();
            String b = in.next();

            // we need to put a good regex equation where in we need to capture the occurence of each AlPhabet
            // we need to get the specifc alphabet from the string and club each recurence into another string
            // incase there is only one occurence we will just say "NO" since the lady bug would not be happy.

            // we need to append the _ to last of the string

            // Selection sort which would be placcing the similar colored
            // bugs in adjecent to each other.

            // Or I can investigate the Hash Data Structure.

            // Once Sorted we will re run through the array to check adjeceny of each color, Even if one does
            // not match print NO or else print YES



            //Another Approach
            // ArrayList which is a 2 diminsion array, and  dimension would like this
             // Row = Alphabets + _
             // coloums = N;
             //  if the colomn for each letter has more than one then we will print Yes else No


            // Have array List add the first exnountered letter as first element, then
            // then next element would be the position of repetition
        }
    }
}

```

Line: 39 Col: 30

Upload Code as File □ Test against custom input Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature