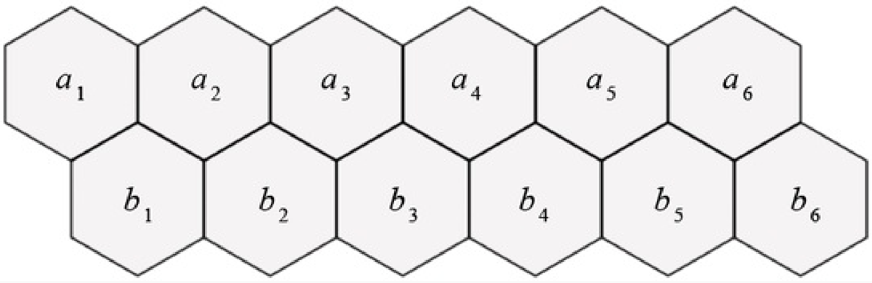
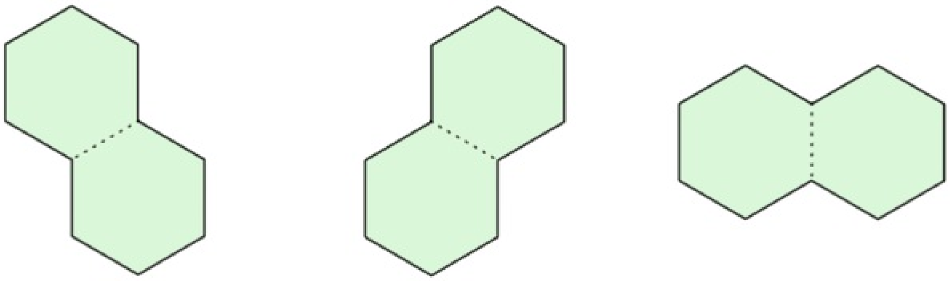
You are given a hexagonal grid consisting of two rows, each row consisting of n cells. The cells of the first row are labelled $a_1, a_2, \ldots a_n$ and the cells of the second row are labelled $b_1, b_2, \ldots, b_n$.

For example, for $n = 6$:



(Note that the b_i is connected with a_{i+1}.)

Your task is to tile this grid with 2×1 *tiles* that look like the following:

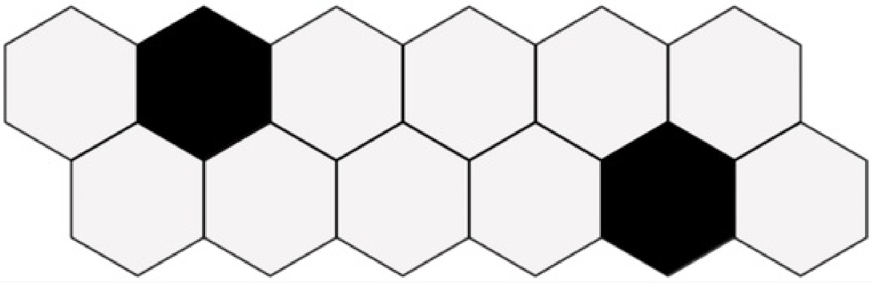


As you can see above, there are three possible orientations in which a tile can be placed.

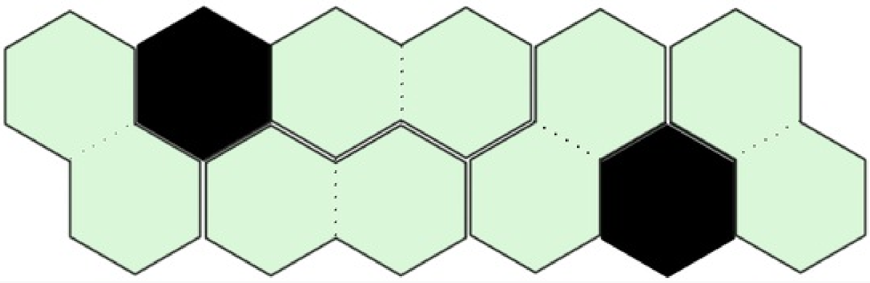
Your goal is to tile the whole grid such that every cell is covered by a tile, and no two tiles occupy the same cell. To add to the woes, certain cells of the hexagonal grid are *blackened*. No tile must occupy a blackened cell.

Is it possible to tile the grid?

Here's an example. Suppose we want to tile this grid:



Then we can do the tiling as follows:



Input Format

The first line contains a single integer t, the number of test cases.

The first line of each test case contains a single integer n denoting the length of the grid.
The second line contains a binary string of length n. The i^{th} character describes whether cell a_i is blackened.
The third line contains a binary string of length n. The i^{th} character describes whether cell b_i is blackened.
A `0` corresponds to an empty cell and a `1` corresponds to blackened cell.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n \leq 10$

Output Format

For each test case, print `YES` if there exists at least one way to tile the grid, and `NO` otherwise.

Sample Input 0

```
6
6
010000
000010
2
00
00
2
00
10
2
00
01
2
00
11
2
10
00
```

Sample Output 0

```
YES
YES
NO
NO
YES
NO
```

Explanation 0

The first test case in the sample input describes the example given in the problem statement above.
For the second test case, there are two ways to fill it: either place two diagonal tiles side-by-side or place two horizontal tiles.