10. Unite 4

Program Name: Unite4.java **Input File: unite4.dat**

You are excited to learn your parents have returned from the toy store with a game for you. You are not so excited to find it is called "Unite 4", and is a generic knock-off of a popular game. You are also not so thrilled to discover that there are many pieces missing, so much so that the game is unplayable. Luckily, the rules are intact, so you decide to write a program that simulates the game.

"Unite 4" is similar to tic-tac-toe and is played on a vertical board consisting of 42 holes in 7 columns and 6 rows. One player has red discs and the other has black discs. Players drop their discs in the top of the board, which drop straight down, stacking on one another. The object of the game is to get four of your colored discs in a row: either horizontally, vertically, or diagonally.

For example, consider a game where the following moves were made:
Red column 4, Black column 5, Red column 3, Black column 4, Red column 1, Black column 4, Red column 2

The board would look like the following after each of the moves (with 'R' representing red-colored discs, 'B' representing black-colored discs , 'O' (a capital letter 'o') representing an empty hole, and rows and columns numbered):

```
Red 4      Black 5    Red 3      Black 4    Red 1      Black 4    Red 2
 1234567    1234567    1234567    1234567    1234567    1234567    1234567
1OOOOOOO   1OOOOOOO   1OOOOOOO   1OOOOOOO   1OOOOOOO   1OOOOOOO   1OOOOOOO
2OOOOOOO   2OOOOOOO   2OOOOOOO   2OOOOOOO   2OOOOOOO   2OOOOOOO   2OOOOOOO
3OOOOOOO   3OOOOOOO   3OOOOOOO   3OOOOOOO   3OOOOOOO   3OOOOOOO   3OOOOOOO
4OOOOOOO   4OOOOOOO   4OOOOOOO   4OOOOOOO   4OOOOOOO   4OOOBOOO   4OOOBOOO
5OOOOOOO   5OOOOOOO   5OOOOOOO   5OOOBOOO   5OOOBOOO   5OOOBOOO   5OOOBOOO
6OOORWOO   6OOORBOO   6OORRBOO   6OORRBOO   6RORRBOO   6RORRBOO   6RRRRBOO
```

At this point the game would end as the player with the red discs has four in a row.

Input

- The first line will be a single integer `n` that indicates the number of data sets in the input.
- Each data set will consist of:
  - A line with a single integer `m` that indicates the total number of moves, 1 <= `m` <= 42.
  - The next line will contain a space-separated list of `m` integers, representing alternating moves by the players, with the player with the red discs always going first. Each move will be an integer `c`, 1 <= `c` <= 7, that represents into which column the player drops their disc. Note that all moves will be valid (a player will not drop their disc into a column that is completely full).

Output

For each data set in the input, output the final layout of the board after applying all the moves in the input, as shown in the above example. On the line after the board layout, output "`RED WINS`" if the player with the red discs has won, "`BLACK WINS`" if the player with the black discs has won, or "`NO WINNER YET`", if neither have won. Note that if there is a winner, there will not be any moves after a player has won.

Example Input File

```
4
7
4 5 3 4 1 4 2
11
4 5 5 6 6 7 6 7 7 4 7
21
```

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

Example Output to Screen

```
 1234567
1OOOOOOO
2OOOOOOO
3OOOOOOO
4OOOBOOO
5OOOBOOO
6RRRRBOO
RED WINS
 1234567
1OOOOOOO
2OOOOOOO
3OOOOOOR
4OOOOORR
5OOOBRRB
6OOORBBB
RED WINS
 1234567
1OOOOOOO
2OOOOOOO
3OOOOOOO
4RBRBRBR
5BRBRBRB
6RBRBRBR
NO WINNER YET
 1234567
1OOOOOOO
2OOOOOOO
3OOOOOOB
4OOOOOOB
5OORROOOB
6ORRROOB
BLACK WINS
```