Grid Challenge

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);

/*
 * Complete the 'gridChallenge' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts STRING_ARRAY grid as parameter.
 */

string gridChallenge(vector<string> grid) {
    for(int i=0;i<grid.size();i++){
        sort(grid[i].begin(),grid[i].end());
    }
    int n=grid.size();
    int m=grid[0].size();

    for(int c=0;c<m;c++){
        for(int r=0;r<n-1;r++){
            if(grid[r][c]>grid[r+1][c]){
                return "NO";
            }
        }
    }
    return "YES";
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string t_temp;
    getline(cin, t_temp);

    int t = stoi(ltrim(rtrim(t_temp)));

    for (int t_itr = 0; t_itr < t; t_itr++) {
        string n_temp;
        getline(cin, n_temp);
```

```
        int n = stoi(ltrim(rtrim(n_temp)));

        vector<string> grid(n);

        for (int i = 0; i < n; i++) {
            string grid_item;
            getline(cin, grid_item);

            grid[i] = grid_item;
        }

        string result = gridChallenge(grid);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}
```