The Bomberman Game

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'bomberMan' function below.
 *
 * The function is expected to return a STRING_ARRAY.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. STRING_ARRAY grid
 */

vector<string> explode(vector<string> grid, int r, int c) {
    vector<string>newGrid(r,string(c,'O'));
    for (int i=0;i<r;i++) {
        for (int j=0;j<c;j++) {
            if (grid[i][j]=='O') {
                newGrid[i][j]='.';
                if(i>0)newGrid[i-1][j]='.';
                if(i<r-1)newGrid[i + 1][j]='.';
                if(j>0)newGrid[i][j-1]='.';
                if(j<c-1)newGrid[i][j+1]='.';
            }
        }
    }
    return newGrid;
}

vector<string> bomberMan(int n, vector<string> grid) {
    int r=grid.size();
    int c=grid[0].size();
    if (n==0||n==1){
        return grid;
    }
    if(n%2==0){
        return vector<string>(r,string(c,'O'));
    }
    vector<string> firstExplosion=explode(grid,r,c);
```

```
    if(n%4==3){
        return firstExplosion;
    }
    return explode(firstExplosion,r,c);
}

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

    string first_multiple_input_temp;
    getline(cin, first_multiple_input_temp);

    vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));

    int r = stoi(first_multiple_input[0]);

    int c = stoi(first_multiple_input[1]);

    int n = stoi(first_multiple_input[2]);

    vector<string> grid(r);

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

        grid[i] = grid_item;
    }

    vector<string> result = bomberMan(n, grid);

    for (size_t i = 0; i < result.size(); i++) {
        fout << result[i];

        if (i != result.size() - 1) {
            fout << "\n";
        }
    }

    fout << "\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;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push back(str.substr(start));

    return tokens;
}
```