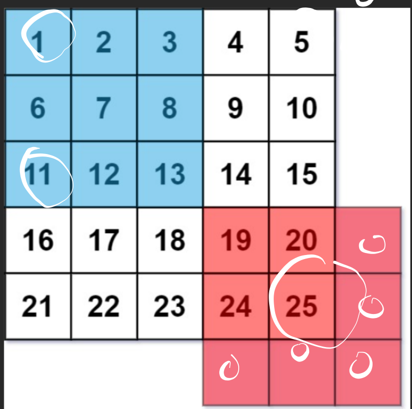
Example 2:

100	200	100
200	50	200
100	200	100

137	141	137
141	138	141
137	141	137

An **image smoother** is a filter of the size `3 x 3` that can be applied to each cell of an image by rounding down the average of the cell and the eight surrounding cells (i.e., the average of the nine cells in the blue smoother). If one or more of the surrounding cells of a cell is not present, we do not consider it in the average (i.e., the average of the four cells in the red smoother).



Given an `m x n` integer matrix `img` representing the grayscale of an image, return *the image after applying the smoother on each cell of it.*

j-1 < 0   j+1 > n

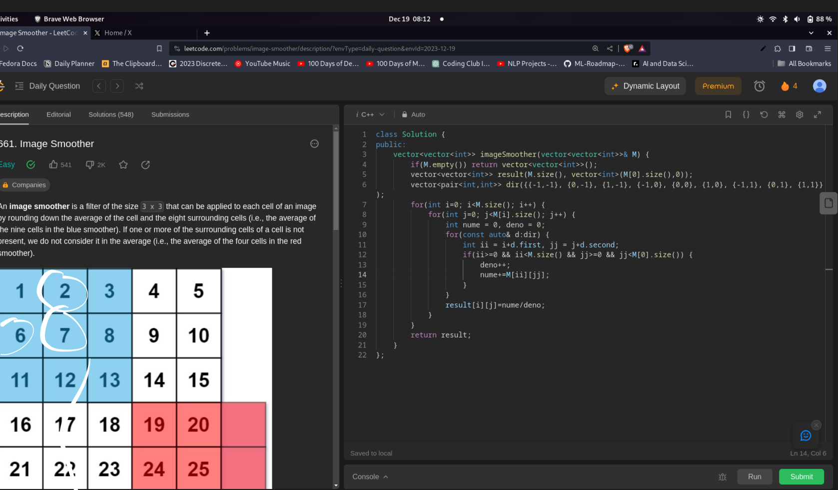
① temp[m][n]

for(i=0; i<m; i++)
{
 for(j=0; j<n; j++)
 { if(j-1 > 0)
 → j+1 i+1
 [i+1][j+1]

② j-1 ≥ 0

↳ temp[m][n]
= arr[i][j+1]
+ arr[i-1][j]
+ arr[i-1][j+1]
+ arr[i+1][j] +
arr[i+1][j+1]

arr[i][j+1] + arr[i][j-1]
+ arr[i-1][j-1] + arr[i-1][j] +
arr[i-1][j+1] + arr[i+1][j-1] +
arr[i+1][j] + arr[i+1][j+1]




```cpp
class Solution {
public:
    vector<vector<int>> imageSmoother(vector<vector<int>>& M) {
        if(M.empty()) return vector<vector<int>>();
        vector<vector<int>> result(M.size(), vector<int>(M[0].size(),0));
        vector<pair<int,int>> dir({{-1,-1}, {0,-1}, {1,-1}, {-1,0}, {0,0}, {1,0}, {-1,1}, {0,1}, {1,1}});
        for(int i=0; i<M.size(); i++) {
            for(int j=0; j<M[i].size(); j++) {
                int nume = 0, deno = 0;
                for(const auto& d:dir) {
                    int ii = i+d.first, jj = j+d.second;
                    if(ii>=0 && ii<M.size() && jj>=0 && jj<M[0].size()) {
                        deno++;
                        nume+=M[ii][jj];
                    }
                }
                result[i][j]=nume/deno;
            }
        }
        return result;
    }
};
```