Coloring Border in C++

```
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int>> dirs = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

void dfs(vector<vector<int>>& grid, int row, int col, int
clr) {
    grid[row][col] = -clr;
    int count = 0;

    for (auto dir : dirs) {
        int rowdash = row + dir[0];
        int coldash = col + dir[1];

        if (rowdash < 0 || coldash < 0 || rowdash >=
grid.size() || coldash >= grid[0].size() ||
abs(grid[rowdash][coldash]) != clr) {
            continue;
        }

        count++;

        if (grid[rowdash][coldash] == clr) {
            dfs(grid, rowdash, coldash, clr);
        }
    }

    if (count == 4) {
        grid[row][col] = clr;
    }
}

void coloring_border(vector<vector<int>>& grid, int row,
int col, int color) {
    dfs(grid, row, col, grid[row][col]);

    for (int i = 0; i < grid.size(); i++) {
        for (int j = 0; j < grid[0].size(); j++) {
            if (grid[i][j] < 0) {
                grid[i][j] = color;
            }
        }
    }
}

int main() {
    // Hardcoded input
    int m = 4;
    int n = 4;
    vector<vector<int>> arr = {
        {2, 1, 3, 4},
        {1, 2, 2, 2},
        {3, 2, 2, 2},
        {1, 2, 2, 2}
    };
    int row = 1;
    int col = 1;
    int color = 3;
```

Input:

```
grid = {
    {2, 1, 3, 4},
    {1, 2, 2, 2},
    {3, 2, 2, 2},
    {1, 2, 2, 2}
}
start = (1, 1)
color = 3
```

🧭 Initial Color at (1, 1): `2`

🔁 DFS Dry Run (Marking Border)

Step	Cell	Action	Count of Same Color Neighbors	Final Cell State
1	(1,1)	Mark `-2`, recurse	0 → Recursing neighbors	-2
2	(1,2)	Mark `-2`, recurse	0 → Recursing	-2
3	(1,3)	Mark `-2`, recurse	0 → Recursing	-2
4	(2,3)	Mark `-2`, recurse	0	-2
5	(2,2)	Mark `-2`, recurse	1	-2
6	(2,1)	Mark `-2`, recurse	2	-2
7	(3,1)	Mark `-2`, recurse	0	-2
8	(3,2)	Mark `-2`, recurse	1	-2
9	(3,3)	Mark `-2`, recurse	1	-2

Once recursion returns, it checks `count == 4`. If true, the cell is fully surrounded by the same component → restore it to `2`. Otherwise, it's on border → leave as `-2`.

Only cell (2,2) has all 4 neighbors of same component → reset to `2`.

🖌 Coloring Step:

- Any cell still marked as `-2` → set to

```
    coloring_border(arr, row, col, color);

    // Print the modified grid
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout << arr[i][j] << "\t";
        }
        cout << endl;
    }

    return 0;
}
```

new `color = 3`

✅ Final Output Grid:

```
2       1       3       4
1       3       3       3
3       3       2       3
1       3       3       3
```

📜 Dry Run Summary Table (Key Points):

Cell	Was Visited	Final Value
(1,1)	✅	3
(1,2)	✅	3
(1,3)	✅	3
(2,1)	✅	3
(2,2)	✅	2
(2,3)	✅	3
(3,1)	✅	3
(3,2)	✅	3
(3,3)	✅	3

Output:-

```
2       1       3       4
1       3       3       3
3       3       2       3
1       3       3       3
```