Given a matrix of ‘O’ and ‘X’, replace ‘O’ with ‘X’ if surrounded by ‘X’

```
/* Novice level coding problem */
```

GLIDER

Problem :

Given a matrix where every element is either ‘O’ or ‘X’, we want to replace the ‘O’s with ‘X’, if the ‘O’s are surrounded by ‘X’. Let’s see an example to understand what “surrounded” means.

‘X’	‘X’	‘X’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘O’	‘O’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘X’	‘O’	‘O’

‘X’	‘X’	‘X’	‘X’
‘X’	‘X’	‘X’	‘X’
‘X’	‘X’	‘X’	‘X’
‘X’	‘X’	‘X’	‘X’
‘X’	‘X’	‘O’	‘O’

But what does “surrounded” exactly mean? We say that a ‘O’, or a set of ‘O’s, is considered to be surrounded by ‘X’ if there are ‘X’ at locations just below, above, left and right of that ‘O’ or set of ‘O’s. Let’s see examples where the ‘O’s are considered as surrounded (green) and not surrounded (red) by ‘X’.

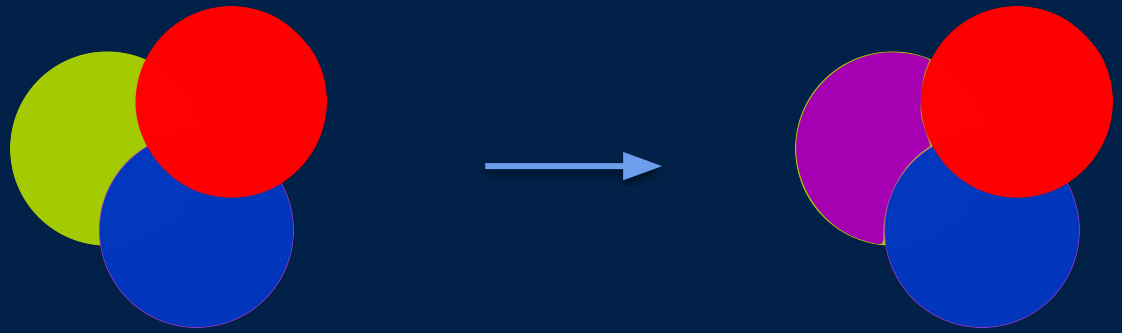
‘X’	‘X’	‘X’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘O’	‘O’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘X’	‘O’	‘O’

As a general rule we can state that if the ‘O’ or group of ‘O’s lies at the edge of the matrix, they can’t be surrounded by ‘X’.

In the example, the two ‘O’s in red don’t have ‘X’ below or to the right of them.

‘X’	‘X’	‘X’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘O’	‘O’	‘X’
‘X’	‘O’	‘X’	‘X’
‘X’	‘X’	‘O’	‘O’

To make an algorithm to solve this problem, we need to use a function called “Flood Fill”. This function is used by programs like MS-Paint. When we take the brush tool and click on one pixel of an image, the color of the region surrounded by that pixel is replaced with a new selected color.



Selecting the yellow circle, we only change that color.

Here, the selected yellow color changed , and the others didn’t.

As an example, let's say that the image is a 2D array, and the colors are given by numbers 1 and 2. If the new color is 3, and we "paint" the cell that has a red circle, we obtain the following:

1	1	2	1
1	2	1	1
1	1	2	1
2	1	2	1
1	1	1	2

3	3	2	1
3	2	1	1
3	3	2	1
2	3	2	1
3	3	3	2

Let's see a possible code to do this task.

Code:

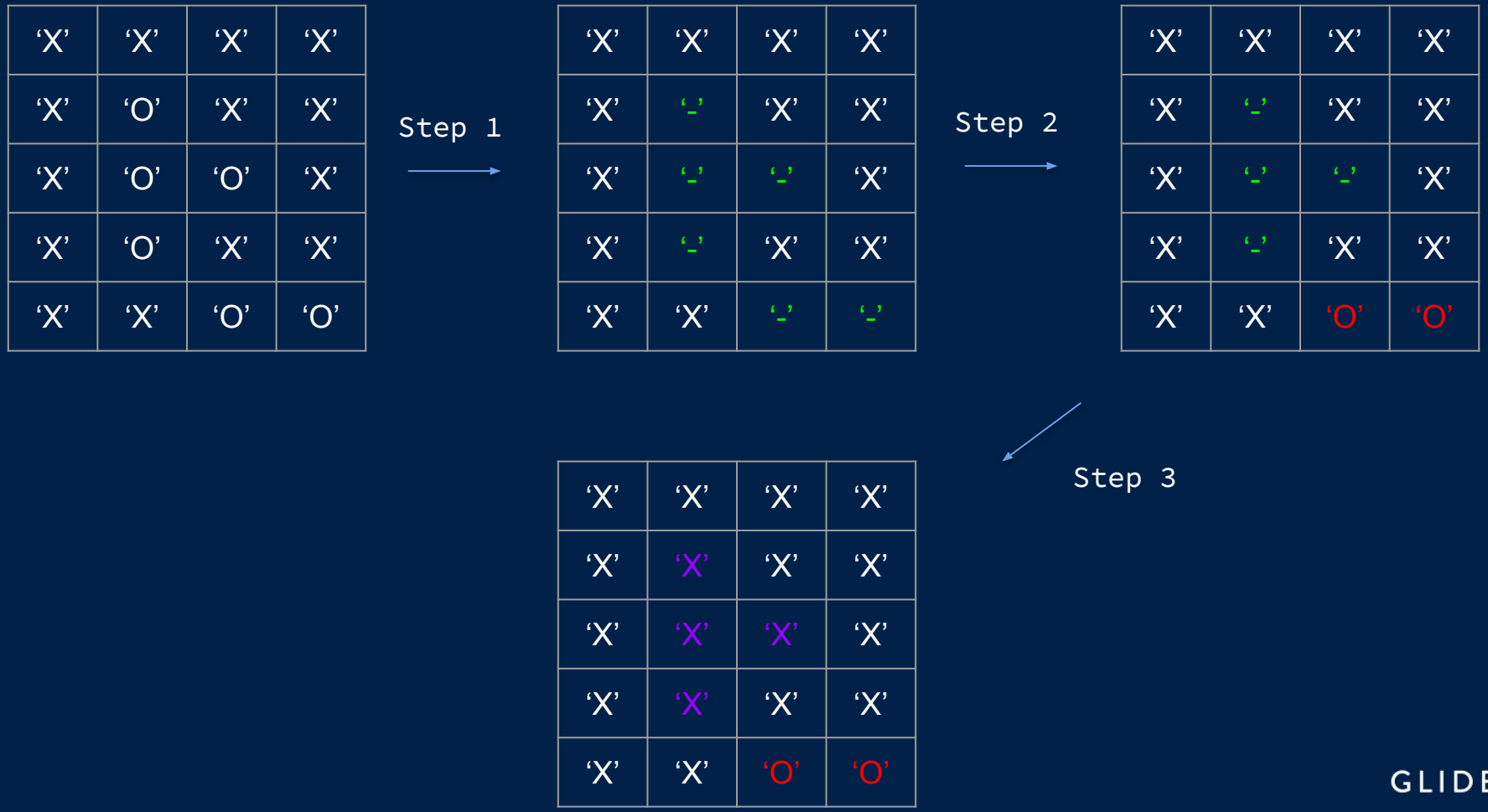
```
def floodFill(screen, x, y, newC):
    prevC = screen[x][y]
    return floodFillUtil(screen, x, y, prevC, newC)

def floodFillUtil(screen, x, y, prevC, newC):
    n, m = len(screen), len(screen)[0]
    if (x<0 or x>n-1 or y<0 or y>m-1): return screen
    if (screen[x][y] != prevC): return screen
    screen[x][y] = newC
    floodFillUtil(screen, x+1, y, prevC, newC)
    floodFillUtil(screen, x-1, y, prevC, newC)
    floodFillUtil(screen, x, y+1, prevC, newC)
    floodFillUtil(screen, x, y-1, prevC, newC)
    return screen
```

Now let’s see how can we solve our original problem using the Flood Fill algorithm. The idea is that we want to use that same idea, but keeping in mind that the cases that are in the boundary of the array are not contemplated.

`// Step 1:` The first step in the algorithm is to replace every element of the matrix that contains an ‘O’ with the character ‘-’.

`// Step 2:` Now we traverse the four edges of the given matrix, and call the function floodFill, to replace the elements containing ‘-’s with ‘O’s.

`// Step 3:` Finally, traverse all elements of the matrix and replace the remaining ‘-’s with ‘X’s. This way we end up with the desired result.


‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘O’ ‘X’ ‘X’
‘X’ ‘O’ ‘O’ ‘X’
‘X’ ‘O’ ‘X’ ‘X’
‘X’ ‘X’ ‘O’ ‘O’
Step 1
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘-’ ‘X’ ‘X’
‘X’ ‘-’ ‘-’ ‘X’
‘X’ ‘-’ ‘X’ ‘X’
‘X’ ‘X’ ‘-’ ‘-’
Step 2
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘-’ ‘X’ ‘X’
‘X’ ‘-’ ‘-’ ‘X’
‘X’ ‘-’ ‘X’ ‘X’
‘X’ ‘X’ ‘O’ ‘O’
Step 3
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘X’ ‘X’ ‘X’
‘X’ ‘X’ ‘O’ ‘O’

GLIDER

Code:

```
def replaceSurrounded(mat):
    n, m = len(screen), len(screen)[0]
    for i in range(n):
        for j in range(m):
            if mat[i][j] == ‘O’: mat[i][j] = ‘-’
    for j in range(m):
        if mat[0][j] == ‘-’: mat = floodFill(mat, 0, j, ‘O’)
    for i in range(n):
        if mat[i][m-1] == ‘-’: mat = floodFill(mat, i, m-1, ‘O’)
    for j in range(m):
        if mat[n-1][j] == ‘-’: mat = floodFill(mat, n-1, j, ‘O’)
    for i in range(n):
        if mat[i][0] == ‘-’: mat = floodFill(mat, i, 0, ‘O’)
```

Code:

```
def replaceSurrounded(mat):


    for i in range(n):
        for j in range(m):
            if mat[i][j] == ‘-’: mat[i][j] = ‘X’

    return mat
```