2326. Spiral Matrix IV

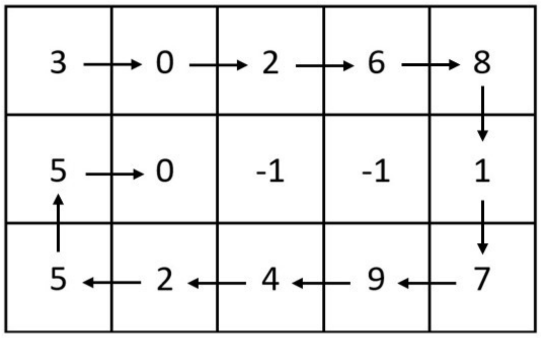
You are given two integers `m` and `n`, which represent the dimensions of a matrix.

You are also given the `head` of a linked list of integers.

Generate an `m x n` matrix that contains the integers in the linked list presented in **spiral** order **(clockwise)**, starting from the **top-left** of the matrix. If there are remaining empty spaces, fill them with `-1`.

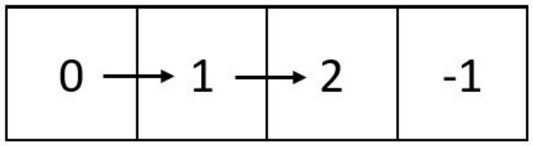
Return *the generated matrix*.

Example 1:



```
Input: m = 3, n = 5, head = [3,0,2,6,8,1,7,9,4,2,5,5,0]
Output: [[3,0,2,6,8],[5,0,-1,-1,1],[5,2,4,9,7]]
Explanation: The diagram above shows how the values are
printed in the matrix.
Note that the remaining spaces in the matrix are filled
with -1.
```

Example 2:



```
Input: m = 1, n = 4, head = [0,1,2]
Output: [[0,1,2,-1]]
Explanation: The diagram above shows how the values are
printed from left to right in the matrix.
The last space in the matrix is set to -1.
```

Constraints:

- `1 <= m, n <= 10`5
- `1 <= m * n <= 10`5
- The number of nodes in the list is in the range `[1, m * n]`.
- `0 <= Node.val <= 1000`

2326. Spiral Matrix IV

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
```

2326. Spiral Matrix IV

```
/*
 Time complexity: O(L*(m+n))
 Space complexity: O(1)
 L: size of the linked list
 m: number of rows
 n: number of columns
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class Solution {
public:
    vvi spiralMatrix(int m, int n, ListNode* head){
        int top=0,right=n-1,bottom=m-1,left=0;
        vvi ans(m,vi(n,-1));
        bool ok=left<=right && top<=bottom;
        ListNode* trav=head;
        while(trav&&ok){
            if(ok){
                for(int i=left;trav!=nullptr && i<=right;++i){
                    ans[top][i]=trav->val;
                    trav=trav->next;
                }
                top++;
                for(int i=top;trav!=nullptr && i<=bottom;++i){
                    ans[i][right]=trav->val;
                    trav=trav->next;
                }
                right--;
            }
            ok=left<=right && top<=bottom;
            if(ok){
                for(int i=right;trav!=nullptr && i>=left;--i){
                    ans[bottom][i]=trav->val;
                    trav=trav->next;
                }
                bottom--;
                for(int i=bottom;trav!=nullptr && i>=top;--i){
                    ans[i][left]=trav->val;
                    trav=trav->next;
                }
                left++;
            }
        }
        return ans;
    }};
```