SNo.	Problem Statement
1.	**Medium Level-Maximum size rectangle binary sub-matrix with all 1s.** **Code:**

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

using namespace std;



#define R 4
#define C 4




int maxHist(int row[])
{

    stack<int> res;

    int tval;

    int max_area = 0;

    int area = 0;
    int i = 0;
    while (i < C) {

        if (res.empty() || row[res.top()] <= row[i])
            res.push(i++);

        else {

            tval = row[res.top()];
            res.pop();
            area = tval * i;

            if (!res.empty())
```

```
            area = tval * (i - res.top() - 1);
        max_area = max(area, max_area);
      }
   }


   while (!res.empty()) {
      tval = row[res.top()];
      res.pop();
      area = tval * i;
      if (!res.empty())
         area = tval * (i - res.top() - 1);

      max_area = max(area, max_area);
   }
   return max_area;
}


int maxRectangle(int A[][C])
{

   int res = maxHist(A[0]);


   for (int i = 1; i < R; i++) {

      for (int j = 0; j < C; j++)


         if (A[i][j])
            A[i][j] += A[i - 1][j];


      res = max(res, maxHist(A[i]));
   }

   return res;
}
```

```
int main()
{
   int A[][C] = {
      { 0, 1, 1, 0 },
      { 1, 1, 1, 1 },
      { 1, 1, 1, 1 },
      { 1, 1, 0, 0 },
   };

   cout << "Area of maximum rectangle is "
        << maxRectangle(A);

   return 0;
}
```

2. **Medium Level:Find the number of islands**

Code:

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

using namespace std;

void dfs(vector<vector<int>>&mat,int i,int j,int r,int c)
{
   if(i<0 || j<0 || i>(r-1) || j>(c-1) || mat[i][j]!=1)
   {
      return;
   }
   if(mat[i][j]==1)
   {
      mat[i][j]=0;
      dfs(mat,i+1,j,r,c);
      dfs(mat,i-1,j,r,c);
       dfs(mat,i,j+1,r,c);
        dfs(mat,i,j-1,r,c);
         dfs(mat,i-1,j-1,r,c);
          dfs(mat,i+1,j+1,r,c);
           dfs(mat,i-1,j+1,r,c);
            dfs(mat,i+1,j-1,r,c);
   }
```

```
}

int countIslands(vector<vector<int>> &mat)
{
   int r = mat.size();
   int c = mat[0].size();
   int cnt = 0;
   for (int i = 0; i < r; i++)
   {
      for (int j = 0; j < c; j++)
      {
         if (mat[i][j] == 1)
         {
            mat[i][j] = 0;
            cnt++;
            dfs(mat, i + 1, j, r, c);
            dfs(mat, i - 1, j, r, c);
            dfs(mat, i, j + 1, r, c);
            dfs(mat, i, j - 1, r, c);
            dfs(mat, i + 1, j + 1, r, c);
            dfs(mat, i - 1, j - 1, r, c);
            dfs(mat, i + 1, j - 1, r, c);
            dfs(mat, i - 1, j + 1, r, c);
         }
      }
   }
   return cnt;
}
int main()
{
   vector<vector<int>> mat = {{1, 1, 0, 0, 0},
                              {0, 1, 0, 0, 1},
                              {1, 0, 0, 1, 1},
                              {0, 0, 0, 0, 0},
                              {1, 0, 1, 0, 1}};

   cout << "Number of islands is: " << countIslands(mat);
   return 0;
}
```

3.	**Medium Level:Given a matrix of 'O' and 'X', replace 'O' with 'X' if surrounded by 'X'**
	Code:
	#include <bits/stdc++.h>
	#include <iostream>
	
	using namespace std;
	#define M 6
	#define N 6
	void flood(char mat[][N],int x,int y,char pre,char newP)
	{
	if (x < 0 || x >= M || y < 0 || y >= N)
	return;
	if (mat[x][y] != pre)
	return;
	
	mat[x][y] = newP;
	
	flood(mat, x+1, y, pre, newP);
	flood(mat, x-1, y, pre, newP);
	flood(mat, x, y+1, pre, newP);
	flood(mat, x, y-1, pre, newP);
	}
	
	int replace(char mat[][N])
	{
	for (int i=0; i<M; i++)
	for (int j=0; j<N; j++)
	if (mat[i][j] == 'O')
	mat[i][j] = '-';
	
	for (int i=0; i<M; i++)
	if (mat[i][0] == '-')
	flood(mat, i, 0, '-', 'O');
	for (int i=0; i<M; i++)
	if (mat[i][N-1] == '-')

```
        flood(mat, i, N-1, '-', 'O');
    for (int i=0; i<N; i++)
      if (mat[0][i] == '-')
        flood(mat, 0, i, '-', 'O');
    for (int i=0; i<N; i++)
      if (mat[M-1][i] == '-')
        flood(mat, M-1, i, '-', 'O');


    for (int i=0; i<M; i++)
      for (int j=0; j<N; j++)
        if (mat[i][j] == '-')
            mat[i][j] = 'X';

}
int main()
{
   char mat[][N] =  {{'X', 'O', 'X', 'O', 'X', 'X'},
                    {'X', 'O', 'X', 'X', 'O', 'X'},
                    {'X', 'X', 'X', 'O', 'X', 'X'},
                    {'O', 'X', 'X', 'X', 'X', 'X'},
                    {'X', 'X', 'X', 'O', 'X', 'O'},
                    {'O', 'O', 'X', 'O', 'O', 'O'},
                   };

      replace(mat);
      for (int i=0; i<M; i++)
   {
     for (int j=0; j<N; j++)
        cout << mat[i][j] << " ";
     cout << endl;
   }


   return 0;
}
```

4. **Medium Level:Spiral Matrix**

Code:

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

```
#define M 3
#define N 3
using namespace std;
 vector<int> spiralOrder(vector<vector<int>>& matrix) {

        int T,B,L,R,dir;
        T=0;
        B=matrix.size()-1;
        L=0;
        R=matrix[0].size()-1;
        dir=0;

       vector<int>res;
       while(T<=B and L<=R)
       {
          if(dir==0)
          {
             for(int i=L;i<=R;i++)
                res.push_back(matrix[T][i]);
             T++;
          }
          else if(dir==1)
          {
             for(int i=T;i<=B;i++)
                res.push_back(matrix[i][R]);
             R--;
          }
         else if(dir==2)
          {
             for(int i=R;i>=L;i--)
                res.push_back(matrix[B][i]);
             B--;
          }
           else if(dir==3)
          {
             for(int i=B;i>=T;i--)
                res.push_back(matrix[i][L]);
             L++;
          }
          dir=(dir+1)%4;
```

```
        }
        return res;

    }
int main()
{
  vector<vector<int>>matrix={{1,2,3},{4,5,6},{7,8,9}};
      for (int x : spiralOrder(matrix))
    {
        cout << x << " ";
    }


    return 0;
}
```

5. **Medium Level:Rotate Image**

Code:

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

#define N 4
using namespace std;
void rotate(int arr[N][N])
{

    for (int j = 0; j < N; j++)
    {
        for (int i = N - 1; i >= 0; i--)
            cout << arr[i][j] << " ";
        cout << '\n';
    }
}


int main()
{
    int arr[N][N] = { { 1, 2, 3, 4 },
                      { 5, 6, 7, 8 },
                      { 9, 10, 11, 12 },
                      { 13, 14, 15, 16 } };
```

```
    rotate(arr);
    return 0;
}
```