SNo.	Problem Statement

1.	**Medium Level-Subarray Sums Divisible by K**
	Code:
	#include <bits/stdc++.h>
	#include <iostream>
	
	using namespace std;
	
	int subarraysDivByK(vector<int>& A, int K) {
	vector<int> counts(K, 0);
	int sum = 0;
	for(int x: A){
	sum += (x%K + K)%K;
	counts[sum % K]++;
	}
	int result = counts[0];
	for(int c : counts)
	result += (c*(c-1))/2;
	return result;
	}
	int main()
	{
	vector<int>A={ 4, 5, 0, -2, -3, 1};
	int n=A.size();
	int K=5;
	cout<<subarraysDivByK(A,K);
	return 0;
	}

2.	**Medium Level-Find All Duplicates in an Array**
	Code:
	#include <bits/stdc++.h>
	#include <iostream>
	
	using namespace std;
	
	int findalldupl(int a[],int n)
	{
	unordered_map<int,int>m;
	for(int i=0;i<n;i++)

```
    {
        m[a[i]]++;
    }
    for(auto it:m)
    {
        if(it.second>1)
        {
            cout<<it.first<<" ";
        }
    }
    cout<<"\n";
    return 0;
}
int main()
{
    int a[]={4,3,2,7,8,2,3,1};
    int n=sizeof(a)/sizeof(a[0]);
    cout<<findalldupl(a,n);
    return 0;
}
```

3. Medium Level-Container With Most Water

Code:

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

using namespace std;

int maxwater(vector<int>&v)
{
    int left=0;
    int right=v.size()-1;
    int maxarea=0;
    while(left<right){
        int area=min(v[left],v[right])*(right-left);
        maxarea=max(maxarea,area);
        if(v[left]<v[right])
        left++;
        else
        right--;
```

```
    }
    return maxarea;
}
int main()
{
    vector<int>v={1,8,6,2,5,4,8,3,7};
    int n=v.size();
    cout<<maxwater(v);
    return 0;
}
```

4. **3Sum (Brute as well as Optimal)**

Code:

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

 void triplets(int a[],int n){
     /*bool have=false;
     for (int i=0; i<n-2; i++)
    {
        for (int j=i+1; j<n-1; j++)
        {
            for (int k=j+1; k<n; k++)
            {
                if (a[i]+a[j]+a[k] == 0)
                {
                    cout << a[i] << " "<< a[j] << " "<< a[k] <<endl;

                        have = true;
                }
            }
        }
    }*/
    bool have = false;

    for (int i=0; i<n-1; i++)
    {

        unordered_set<int> s;
```

```
        for (int j=i+1; j<n; j++)
        {
            int x = -(a[i] + a[j]);
            if (s.find(x) != s.end())
            {
                printf("%d %d %d\n", x, a[i], a[j]);
                have = true;
            }
            else
                s.insert(a[j]);
        }
    }
    if(have==false)
    cout<<"triplet not exist"<<endl;

}
int main()
{

    int a[] = {0, -1, 2, -3, 1};
    int n = sizeof(a)/sizeof(a[0]);
    triplets(a, n);
    return 0;

}
```

5. **Medium Level-Maximum Points You Can Obtain from Cards**

Code:

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

using namespace std;
int findpoint(int a[],int n,int k)
{
        int sum=0;

        int ans=0;
        for(int i=0;i<k;i++){
            sum+=a[i];
        }
        ans=sum;
```

```
        int i=k-1,j=n-1;
        while(i>=0 && j>=n-k){
            sum-=a[i];
            sum+=a[j];
            i--;
            j--;
            ans=max(sum,ans);
        }
        return ans;
}
int main()
{
    int a[]={1,2,3,4,5,6,1};
    int n=sizeof(a)/sizeof(a[0]);
    int k=3;
    cout<<findpoint(a,n,k);
    return 0;
}
```

6. **Medium Level-Subarray Sum Equals K**

Code:

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

using namespace std;
 int subarraySum(int nums[],int n, int k) {


        int count=0;
        unordered_map<int,int>prevSum;
        int sum=0;
        for(int i=0;i<n;i++){
        sum+=nums[i];
        if(sum==k)
        count++;
        if(prevSum.find(sum-k)!=prevSum.end()){
        count+=prevSum[sum-k];
        }
        prevSum[sum]++;
        }
```

```
    return count;
 }
int main()
{
    int nums[]={1,1,1};
    int n=sizeof(nums)/sizeof(nums[0]);
    int k=2;
    cout<<subarraySum(nums,n,k);
    return 0;
}
```

7. **Medium Level-Spiral Matrix**

Code:

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

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, 10, 11, 12},
                               {13, 14, 15, 16}};
    for(int x:spiralOrder(matrix))
    {
        cout << x << " ";
    }

    return 0;
}
```

8. **Medium Level-Word Search**

Code:

```
bool dfs(vector<vector<char>>& board, string &word,int i,int j){

    //base case
    if(word.size()==0) return true;
    if(i<0 || j<0 || i>=board.size() || j>= board[0].size() ||
board[i][j]!=word[0]) return false;

    char c = board[i][j];
```

```
        board[i][j] ='X';
        string s = word.substr(1);

        //dfs call
        bool res = dfs(board,s,i+1,j)||dfs(board,s,i-
1,j)||dfs(board,s,i,j+1)||dfs(board,s,i,j-1);

        //backtrack
        board[i][j] =c;
        return res;
    }
    bool exist(vector<vector<char>>& board, string word) {
        int m = board.size();
        int n = board[0].size();

        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                if(dfs(board,word,i,j)) return true;
            }
        }
        return false;

    }
```

9. Medium Level-Jump Game

Code:

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

using namespace std;
bool canJump(int a[],int n)
{
    int reach=0;
        for(int i=0;i<n;i++)
        {
            if(reach < i)

                return false;
                reach=max(reach,i+a[i]);
```

```
        }
      return true;
}
int main()
{
   int a[]={2,3,1,1,4};
   int n=sizeof(a)/sizeof(a[0]);
   cout<<canJump(a,n)<<endl;
   return 0;
}
```

10. Medium Level-Merge Sorted Array.

Code:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;


void mergeArrays(int arr1[], int arr2[], int n1,
                 int n2, int arr3[])
{
   int i = 0, j = 0, k = 0;


   while (i<n1 && j <n2)
   {

      if (arr1[i] < arr2[j])
         arr3[k++] = arr1[i++];
      else
         arr3[k++] = arr2[j++];
   }


   while (i < n1)
      arr3[k++] = arr1[i++];


   while (j < n2)
      arr3[k++] = arr2[j++];
}
```

```
int main()
{
   int arr1[] = {1, 3, 5, 7};
   int n1 = sizeof(arr1) / sizeof(arr1[0]);

   int arr2[] = {2, 4, 6, 8};
   int n2 = sizeof(arr2) / sizeof(arr2[0]);

   int arr3[n1+n2];
   mergeArrays(arr1, arr2, n1, n2, arr3);


   for (int i=0; i < n1+n2; i++)
      cout << arr3[i] << " ";

   return 0;
}
```

11. **Medium Level-Majority Element.**

Code:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;


 int majorityElement(vector<int>& nums) {

    unordered_map<int,int>m;
    int n=nums.size();
    for(int i=0;i<nums.size();i++)
    {
       m[nums[i]]++;
       if(m[nums[i]]>(n/2))
       return nums[i];
    }
    return 0;
 }
int main()
{
```

```
    vector<int>nums={3,2,3};
    int n=nums.size();
    cout<<majorityElement(nums);
    return 0;
}
```

12. **Medium Level-Reverse Pairs.**

Code:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
class Solution
{
    public:
     void mergeArray(vector<int> &arr, int low, int mid, int high, int
&cnt)
{

       int l = low, r = mid + 1;
        while(l <= mid && r <= high){
            if((long)arr[l] > (long) 2 * arr[r]){
                cnt += (mid - l + 1);
                r++;
            }else{
                l++;
            }
        }
 sort(arr.begin()+low, arr.begin()+high+1 );
}

void mergeSort(vector<int> &arr, int low, int high, int &cnt)
{
    if (low < high)
    {
        int mid = low + (high - low) / 2;
        mergeSort(arr, low, mid, cnt);
        mergeSort(arr, mid + 1, high,cnt);

        mergeArray(arr, low, mid, high, cnt);
    }
}
```

```
    int reversePairs(vector<int>& arr) {
        int cnt = 0;
         mergeSort(arr, 0, arr.size() - 1, cnt);
        return cnt;

    }
};


int main()
{
    Solution ob;
    vector<int> v = {2,8,7,7,2};
    cout << (ob.reversePairs(v));

}
```

13. **Medium Level-Print all possible combinations of r elements in a given array of size n.**

Code:

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

using namespace std;

void comUtil(int arr[], int n, int r,

                int index, int data[], int i);



void printCom(int arr[], int n, int r)

{



    int data[r];
```

```
    comUtil(arr, n, r, 0, data, 0);
}



void comUtil(int arr[], int n, int r,
                int index, int data[], int i)
{

    if (index == r)
    {
        for (int j = 0; j < r; j++)
            cout << data[j] << " ";
        cout << endl;
        return;
    }



    if (i >= n)
        return;
```

```
    data[index] = arr[i];
    comUtil(arr, n, r, index + 1, data, i + 1);



    comUtil(arr, n, r, index, data, i+1);
}



int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int r = 3;
    int n = sizeof(arr)/sizeof(arr[0]);
    printCom(arr, n, r);
    return 0;
}
```

14. **Medium Level-Game Of Life.**

Code:

```
class Solution {
public:

    int life(vector<vector<int>>& board,int i,int j)
    {
        if(i<0||j<0||i>=board.size()||j>=board[0].size()||board[i][j]==0)
```

```
        {
            return 0;
        }
        return 1;
    }

    int checklive(vector<vector<int>>& board,int i,int j)
    {
        int k=0;

        if(life(board,i-1,j)==1)
        {
            k++;
        }
        if(life(board,i,j-1)==1)
        {
            k++;
        }
        if(life(board,i+1,j+1)==1)
        {
            k++;
        }
        if(life(board,i+1,j)==1)
        {
            k++;
        }
        if(life(board,i-1,j-1)==1)
        {
            k++;
        }
        if(life(board,i,j+1)==1)
        {
            k++;
        }
        if(life(board,i+1,j-1)==1)
        {
            k++;
        }
        if(life(board,i-1,j+1)==1)
        {
```

```
            k++;
        }
        if(board[i][j]==0 and k==3)
        {
            return 1;
        }
        if(board[i][j]==1 and (k==2||k==3))
        {
            return 1;
        }
        return 0;
    }
    void gameOfLife(vector<vector<int>>& board) {


vector<vector<int>>a(board.size(),vector<int>(board[0].size(),0));
        for(int i=0;i<board.size();i++){
            for(int j=0;j<board[0].size();j++){
                a[i][j]=checklive(board,i,j);
            }
        }
        board=a;
    }
};
```