## EASY :

### 1. Find all pairs on array whose sum is equal to given sum

```
class Solution{
public:
   int getPairsCount(int arr[], int n, int k) {

       unordered_map<int,int>mp;
       for(int i=0;i<n;i++)
       mp[arr[i]]++;

       int count=0;

       for(int i=0;i<n;i++)
       {
          mp[arr[i]]--;
          count+=mp[k-arr[i]];
       }
       return count;
   }
};
```

### 2. Common in 3 sorted array

```
class Solution{
public:
   int getPairsCount(int arr[], int n, int k) {

       unordered_map<int,int>mp;
       for(int i=0;i<n;i++)
       mp[arr[i]]++;

       int count=0;

       for(int i=0;i<n;i++)
       {
          mp[arr[i]]--;
          count+=mp[k-arr[i]];
       }
       return count;
   }
};
```

### 3. Array subset of another array

```
string isSubset(int a1[], int a2[], int n, int m)
{
   unordered_map<int,int>mp;
   for(int i=0;i<m;i++)
   mp[a2[i]]++;
   int c=0;
   for(int i=0;i<n;i++)
   {
     if(mp[a1[i]])
      c++;
   }
   if(c==m)
   return "Yes";
   else
   return "No";
}
```

## MEDIUM :

### 1. Pascal triangle

```
class Solution {
public:
   vector<vector<int>> generate(int numRows)
   {
     vector<vector<int>>ans;
       vector<int>ans1;
        ans1.push_back(1);
        ans.push_back(ans1);
       if(numRows==1)
       return ans;
       ans1.push_back(1);
       ans.push_back(ans1);
       if(numRows==2)
       return ans;
       ans1.clear();
       for(int i=2;i<numRows;i++)
       {
          ans1.push_back(1);
         for(int j=1;j<=i;j++)
         {
```

```
            if(i==j)
            ans1.push_back(1);
            else
            {
               int v=ans[i-1][j]+ans[i-1][j-1];
               ans1.push_back(v);
            }
         }
           ans.push_back(ans1);
           ans1.clear();
      }
       return ans;
    }
};
```

**2. Merge sort**

```
void merge(int *Arr, int start, int mid, int end)
{
        int temp[end - start + 1];

        int i = start;
        int j = mid+1;
         int k = 0;

        while(i <= mid && j <= end)
       {
                if(Arr[i] <= Arr[j])
                {
                        temp[k] = Arr[i];
                        k++;
                         i++;
                }
                else
                {
                        temp[k] = Arr[j];
                        k++;
                       j++;
                }
        }

        while(i <= mid)
         {
                temp[k] = Arr[i];
                k++;
```

```
            i++;
        }

        while(j <= end)
        {
            temp[k] = Arr[j];
            k++;
            j++;
        }

        for(i = start; i <= end; i += 1)
        Arr[i] = temp[i - start];

}


void mergeSort(int *Arr, int start, int end)
{
    if(start < end)
    {
            int mid = (start + end) / 2;
            mergeSort(Arr, start, mid);
            mergeSort(Arr, mid+1, end);
            merge(Arr, start, mid, end);
        }
}
```

### 3. Repeat and Missing Number

```
int findDuplicate(vector<int>& nums) {
    unordered_map<int,int> m;
    int res;
    for(int i = 0; i < nums.size(); i++)
    {
        if(m.count(nums[i]) != 0)
        {
            res = nums[i];
            break;
        }
        m[nums[i]] = 1;
    }
    return res;
}
```

### 4. Subarray with sum 0

```
class Solution{
```

```
  public:
  bool subArrayExists(int arr[], int n)
  {
     int sum = 0 ;
     unordered_set<int>st;
     for(int i=0;i<n;i++)
     {
        sum+=arr[i];
        if(arr[i]==0 || sum==0)
        return true;
        else if(st.find(sum)!=st.end())
        return true;
        else
        st.insert(sum);
     }
  return false;
 }
  };
```

**5. Triplet that sum to a given number**

```
class Solution{
  public:
  bool find3Numbers(int A[], int n, int X)
  {

     sort(A,A+n);
     for(int i=0;i<=n-3;i++)
     {
           int l=i+1;
           int r=n-1;
           while(l<r)
           {

              if(A[l]+A[r]+A[i]<X)
              l++;
              else if( A[l]+A[r]+A[i]>X)
              r--;
              else
              return 1;
           }
     }
    return 0;
  }
};
```

**6. 4 sum**

```
class Solution {
```

```
public:
    int fourSumCount(vector<int>& nums1, vector<int>& nums2, vector<int>& nums3, vector<int>&
nums4) {

        int n = nums1.size();
        unordered_map<int,int>track;

        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            track[nums1[i]+nums2[j]]++;
        }

        int ans=0;
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            ans+=track[0-nums3[i]-nums4[j]];
        }
        return ans;
    }
};
```

### 7. Gas station

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost)
    {
        int total=0;
        int cur=0;
        int st=0;
        int n=gas.size();
        for(int i=0;i<n;i++)
        {
            total+=gas[i]-cost[i];
            cur+=gas[i]-cost[i];
            if(cur<0)
            {
                cur=0;
                st=i+1;
            }
        }
        return (total<0)?-1:st;
    }
};
```