SNo.	Problem Statement

1.	**Easy Level : Permute two arrays such that sum of every pair is greater or equal to K.**
	Code:
	#include <bits/stdc++.h>
	#include <iostream>
	
	using namespace std;
	bool permute(int a[],int n, int b[], int m,int k)
	{
	for(int i=0;i<n;i++)
	
	for(int j=i+1;j<m;j++)
	
	if(a[i]+b[j]>=k)
	return true;
	else
	return false;
	
	}
	int main()
	{
	int a[]={2, 1, 3};
	int n=sizeof(a)/sizeof(a[0]);
	int b[]={7, 8, 9};
	int m=sizeof(b)/sizeof(b[0]);
	int k=10;
	if(permute(a,n,b,m,k))
	cout<<"YES";
	else
	cout<<"NO";
	return 0;
	}

2.	**Easy Level:Ceiling in a sorted array.** **Code:**

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

using namespace std;
int findceil(int a[],int low,int high,int x)
{
   int i;
    if(x <= a[low])
       return low;
   for(i = low; i < high; i++)
   {
       if(a[i] == x)
       return i;



       if(a[i] < x && a[i+1] >= x)
       return i+1;
   }
```

```
    return -1;
}
int main()
{
    int a[]={1, 2, 8, 10, 10, 12, 19};
    int n=sizeof(a)/sizeof(a[0]);

    int x=3;
    int p=findceil(a,0,n-1,x);
    if(p==-1)
    cout<<x;
    else
    cout<<x  <<" -> is : "<<  a[p];
    return 0;
}
```

3. **Easy Level : Find a pair with the given difference.**
Code:

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

using namespace std;
```

```
bool findpair(int a[],int n,int diff)


{
   int i=0;
   int j=1;
   while(i<n and j<n)
   {
      if(i!=j and (abs(a[i]-a[j])==diff))
      {
      cout<<a[i]<<" "<<a[j];
      return true;
      }
      else if(abs(a[i]-a[j])<diff)
      {
      j++;
      }
      else
      i++;
   }
   cout << "No such pair";
   return false;
```

```
}

int main()

{

   int a[]={1, 8, 30, 40, 100};

   int n=sizeof(a)/sizeof(a[0]);



   int diff=60;

   findpair(a,n,diff);



   return 0;

}
```

Medium Level Problem

SNo.	Problem Statement

1.	**Medium Level: Check if reversing a sub array make the array sorted.**
	Code:
	
	`#include<bits/stdc++.h>`
	`using namespace std;`
	
	`bool checkReverse(int a[], int n)`
	`{`
	`  if (n == 1)`
	`  return true;`

```
    int i;
    for (i=1; i < n && a[i-1] < a[i]; i++);
    if (i == n)
        return true;


    int j = i;
    while (j < n && a[j] < a[j-1])
    {
        if (i > 1 && a[j] < a[i-2])
            return false;
        j++;
    }

    if (j == n)
        return true;


    int k = j;


    if (a[k] < a[i-1])
      return false;

    while (k > 1 && k < n)
    {
        if (a[k] < a[k-1])
            return false;
        k++;
    }
    return true;
}



int main()
{
    int a[] = {1, 3, 4, 10, 9, 8};
```

```
    int n = sizeof(a)/sizeof(a[0]);
    checkReverse(a, n)? cout << "Yes" : cout << "No";
    return 0;
}
```

3. **Medium Level : Product of Array except itself**

Code:

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


void productArray(int arr[], int n)
{

    if (n == 1) {
        cout << 0;
        return;
    }

    int i, temp = 1;


    int* prod = new int[(sizeof(int) * n)];

    memset(prod, 1, n);


    for (i = 0; i < n; i++) {
        prod[i] = temp;
        temp *= arr[i];
    }


    temp = 1;


    for (i = n - 1; i >= 0; i--) {
        prod[i] *= temp;
        temp *= arr[i];
    }
```

```
    for (i = 0; i < n; i++)
        cout << prod[i] << " ";

    return;
}


int main()
{
    int arr[] = { 10, 3, 5, 6, 2 };
    int n = sizeof(arr) / sizeof(arr[0]);

    productArray(arr, n);
}
```

4. **Medium Level : Make all array elements equal with minimum cost.**

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


int minCostToMakeElementEqual(int a[], int n)
{
    int o;
    if(n%2==1)
    o=a[n/2];

    else
    o=(a[n/2]+a[(n-2)/2])/2;
    int sum=0;
    for(int i=0;i<n;i++)
    sum+=abs(a[i]-o);
    return sum;
}


int main()
```

```
{
    int a[] = { 1, 100, 101 };
   int n = sizeof(a) / sizeof(a[0]);

   cout << (minCostToMakeElementEqual(a, n));
}
```

5. **Medium Level : Find Peak Element**

Code:

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


 int findPeakElement(vector<int>& nums) {

    int left=0,right=nums.size()-1;
    while(left<right)
    {
        int mid=(left+right)/2;
        if(nums[mid]>nums[mid+1])
        right=mid;
        else
        left=mid+1;
    }
    return left;
 }


int main()
{
    vector<int>nums={1,2,3,1};
    int n=nums.size();

    cout<<findPeakElement(nums);
    return 0;
}
```