1) Given an array and a number K where K is smaller than the size of the array. Find the K'th largest element in the given array. Given that all array elements are distinct.

Sample Input: arr[] = { 5, 2, 7, 1, 3}, k = 2

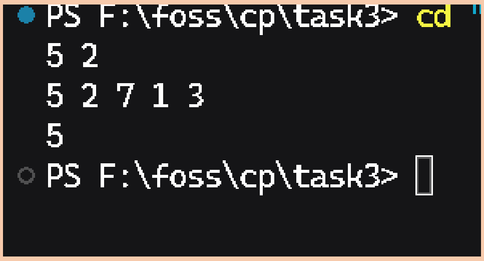
Sample Output: 5

Explanation: The 2nd largest element is 5.

Code:

```
//21161 Shaik Nazeer CSE-B
#include<bits/stdc++.h>
#define ll long long
#define loop(i,n) for(int i = 0; i < n; i++)
#define loop1(i,n) for(int i = 1; i <= n; i++)
using namespace std;
int main()
{
   int n,k;
   cin>>n>>k;
   int a[n];
   loop(i,n) cin>>a[i];
   sort(a,a+n);
   cout<<a[n-k]<<endl;
   return 0;
}
```

Screenshot:



```
● PS F:\foss\cp\task3> cd "
  5 2
  5 2 7 1 3
  5
○ PS F:\foss\cp\task3> ▯
```

2) Given a sorted array arr[] and a number x, write a function that counts the occurrences of x in arr[]. Expected time complexity is O(Logn).

Sample Input: x = 2

A[] = {1,2,2,2,3,4,5,7,9}

Sample Output: 3

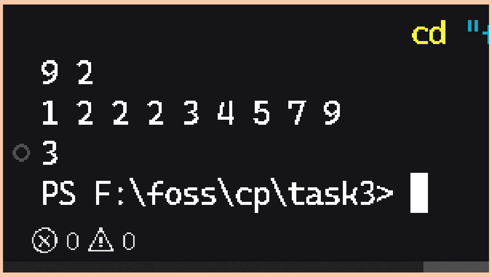
Explanation: The number 2 occurs 3 times in the array.

Code:

```
// 21161 Shaik Nazeer CSE-B
#include <bits/stdc++.h>
#define ll long long
#define loop(i, n) for (int i = 0; i < n; i++)
#define loop1(i, n) for (int i = 1; i <= n; i++)
using namespace std;
int main()
{
   int n,x;
   cin>>n>>x;
   int a[n];
   loop(i,n) cin>>a[i];
   int start = 0, end = n - 1, mid = (start + (end - start) / 2);
   while (start <= end)
   {
      mid = (start + (end - start) / 2);
      if (a[mid] < x)
      {
         start = mid + 1;
      }
      else if (a[mid] > x)
```

```
            {
                end = mid - 1;
            }
            else
            {
                break;
            }
        }
        if (start > end && a[mid] != x)
            return 0;
        int cnt = 0, i = mid + 1;
        while (mid >= 0 && a[mid] == x)
        {
            cnt++;
            mid--;
        }
        while (i < n && a[i] == x)
        {
            i++;
            cnt++;
        }
        cout<<cnt<<endl;
        return 0;
    }
```

Screenshot:



```
                    cd "
9 2
1 2 2 2 3 4 5 7 9
3
PS F:\foss\cp\task3>
```

3) Write a program to reverse an array.

Sample Input: arr[] = {4, 5, 1, 2}

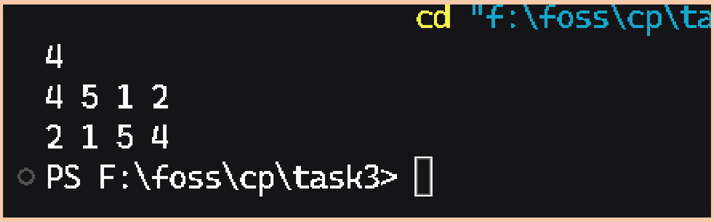
Sample Output: arr[] = {2, 1, 5, 4}.

Explanation: 1100 is the binary representation of the given number 12. Set bit is nothing but the ‘1’ bits that are present in a binary number. In 1100 the first set bit from the right occurs in third position. Thus, the output is 3.

Code:

```
//21161 Shaik Nazeer CSE-B
#include<bits/stdc++.h>
#define ll long long
#define loop(i,n) for(int i = 0; i < n; i++)
#define loop1(i,n) for(int i = 1; i <= n; i++)
using namespace std;
int main()
{
   int n;
   cin>>n;
   int a[n];
   loop(i,n) cin>>a[i];
   reverse(a,a+n);
   loop(i,n) cout<<a[i]<<" ";
   return 0;
}
```

Screenshot:



```
                        cd "f:\foss\cp\ta
4
4 5 1 2
2 1 5 4
PS F:\foss\cp\task3> 
```

4) Given an array Arr[] of N integers. Find the contiguous sub-array (containing at least one number) which has the maximum sum and return its sum. (Kadane’s Algorithm)

Sample Input: a[] = { -2, -3, 4, -1, -2, 1, 5, -3 }

Output: 7

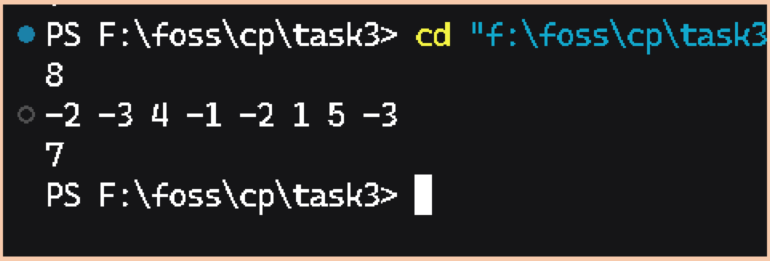
Explanation: The Max subarray sum is 7 of elements (4, -1, -2, 1, 5) which is a contiguous subarray.

Code:

```
//21161 Shaik Nazeer CSE-B
```

```
#include<bits/stdc++.h>
#define ll long long
#define loop(i,n) for(int i = 0; i < n; i++)
#define loop1(i,n) for(int i = 1; i <= n; i++)
using namespace std;
int main()
{
    int n;
    cin>>n;
    int a[n];
    loop(i,n) cin>>a[i];
    int sum = a[0],maxSum = sum;
    loop1(i,n-1){
        if(sum<0){
            sum=0;
        }
        sum+=a[i];
        maxSum=max(sum,maxSum);
    }
    cout<<maxSum<<endl;
    return 0;
}
```

Screenshot:



```
● PS F:\foss\cp\task3> cd "f:\foss\cp\task3
  8
○ -2 -3 4 -1 -2 1 5 -3
  7
  PS F:\foss\cp\task3> █
```

5) **Given an array and a value, find if there is a triplet in array whose sum is equal to the given value. If there is such a triplet present in array, then print the triplet and return true. Else return false.**

Sample Input: arr = {12, 3, 4, 1, 6, 9}, sum = 24;

Sample Output: 12, 3, 9

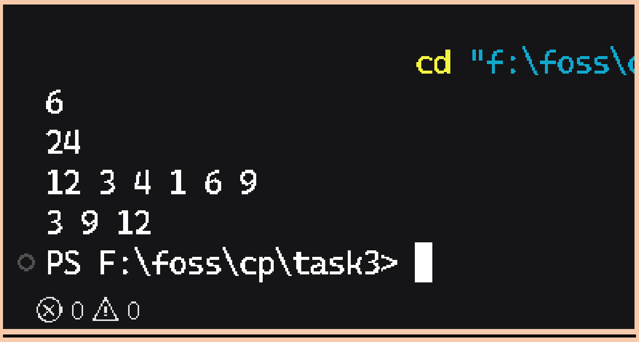
Explanation: There is a triplet (12, 3 and 9) present in the array whose sum is 24.

Constraints: Expected Time Complexity is O(n^2).

Code:

```
// 21161 Shaik Nazeer CSE-B
#include <bits/stdc++.h>
#define ll long long
#define loop(i, n) for (int i = 0; i < n; i++)
#define loop1(i, n) for (int i = 1; i <= n; i++)
using namespace std;
int main()
{
    int n, sum, j, k, tempSum = 0;
    cin >> n >> sum;
    int a[n];
    loop(i, n) cin >> a[i];
    sort(a, a + n);
    loop(i, n)
    {
        tempSum = sum-a[i];
        if(tempSum<0) {
            cout<<"False\n";
            return 0;
        }
        j = i + 1, k = n - 1;
        while (j < k)
        {
            if(tempSum==a[j]+a[k]){
                cout<<a[i]<<" "<<a[j]<<" "<<a[k]<<endl;
                return 0;
            }
            else if(a[j]+a[k]<tempSum){
                j++;
            }else{
                k--;
            }
        }
    }
    return 0;
}
```

Screenshot:



```
                    cd "f:\foss\
6
24
12 3 4 1 6 9
3 9 12
PS F:\foss\cp\task3>
```