13/11/2024
ARIKALEESWARAN G
22AM002

1. Given an array arr[] and an integer k where k is smaller than the size of the array, the task is to find the kth smallest element in the given array.

   Program:

```
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)-> b-a);
        for(int i=0;i<arr.length;i++){
            pq.offer(arr[i]);
            if(pq.size() >k){
                pq.poll();
            }
        }
        return pq.peek();
    }
}
```

   TC : O(*nlog k*)
   SC: O(*k*)

2. Given a sorted array arr and an integer k, find the position(0-based indexing) at which k is present in the array using binary search.

Program:

```
        class Solution {
    public int binarysearch(int[] arr, int k) {
        int l=0;
        int r= arr.length - 1;
        while(l<=r){
            int mid = l + (r-l)/2;
            if(arr[mid] == k){
                return mid;
            }else if(arr[mid] < k){
                l= mid + 1;
            }else{
                r = mid - 1;
            }
        }
        return -1;
            }
}
```

TC: O(*log n*)
SC: O(*1*)

3. Given an array arr[ ] of integers, the task is to find the next greater element for each element of the array in order of their appearance in the array. Next greater element of an element in the array is the nearest element on the right which is greater than the current element.
   If there does not exist next greater of current element, then next greater element for current element is -1. For example, next greater of the last element is always -1.

   Program:

```
public ArrayList<Integer> nextLargerElement(int[] arr) {
        // code here
        ArrayList<Integer> lst = new ArrayList<>();
        int l=0;
        int r=0;
        int n= arr.length;

        for(int i=0;i<n;i++){
            int max = -1;
            for(int j=i + 1;j<n;j++){
                if(arr[j] > arr[i]){
                    max = arr[j];
                        break;
                }
            }

            lst.add(max);
            }
         return lst;
    }
}
```

   TC: O(n^2)
   SC: O(n)

4. Given two arrays a[] and b[], the task is to find the number of elements in the union between these two arrays.The Union of the two arrays can be defined as the set containing distinct elements from both arrays. If there are repetitions, then only one element occurrence should be there in the union.

   Program:

```
class Solution {
    public static int findUnion(int a[], int b[]) {
        // code here
        Set<Integer> st  = new HashSet<>();
        for(int n:a){
            st.add(n);
        }
        for(int n:b){
            st.add(n);
        }
        return st.size();
    }
}
```

   TC: O(N+M)
   SC: O(N+M)

5. Given an array arr of non-negative numbers. The task is to find the first equilibrium point in an array. The equilibrium point in an array is an index (or position) such that the sum of all elements before that index is the same as the sum of elements after it.

Program:

```
class Solution {
    // Function to find equilibrium point in the array.
    public static int equilibriumPoint(int arr[]) {
        int lsum = 0;
        int rsum =0;
        for(int n: arr){
            lsum += n;
        }
        for(int i=0;i<arr.length;i++){
            lsum -= arr[i];
            if(lsum == rsum){
                return i + 1;
            }
            rsum += arr[i];

        }
        return -1;
    }}
```

TC: O(n)
SC: O(1)

6. You are given a string s representing an expression containing various types of brackets: {}, (), and []. Your task is to determine whether the brackets in the expression are balanced. A balanced expression is one where every opening bracket has a corresponding closing bracket in the correct order.

Program:

```
class Solution {
    // Function to check if brackets are balanced or not.
    static boolean isParenthesisBalanced(String s) {
        // code here
        Stack<Character> st = new Stack<>();
        boolean c = false;

        for(char ch : s.toCharArray()){
            if(ch == '(' || ch == '{' || ch == '['){
                st.push(ch);
            }
            else if( ch == ')' || ch == '}' || ch == ']'){
                if(st.isEmpty()) return false;

               if((ch == ')' && st.peek() == '(') ||
               (ch == '}' && st.peek() == '{') ||
               (ch == ']' && st.peek() == '[')){
                   st.pop();
               }else{
                   return false;
               }
            }

        }

        return st.isEmpty();
    }
}
```

TC: O(n)
SC: O(n)