Programming Assignment- 1

Anshika Khare
Section - 9

1) Largest Element in array

```
Public static void main(String[] args){
    int arr[]= new int[]{25,11,3,2,56};
    int max=0;
    for(int i=0; i<arr.length ; i++){
        if (arr[i] > max)
            max =arr[i];
    }
    System.out.printen( max);
}
```

T.C => O(n)
S.C => O(1)

2) Reverse a given array

```
Public Static void main(String[] args){
    int arr[]= new int[]{25,11,3,2,56};
    ~~for (int i=0 ; i~~
    int left =0;
    int right = arr.length-1;
    while(left< right){
        int temp = arr[left];
        arr[left]= arr[right];
        arr[right]= tmp;
        left ++;
        right --;
    }
} ~~SystemoutP~~
```

T.C => O(n)
S.C => O(1)

3) Second Largest Element

```
Public Static int SecondLargest(int[] arr) {
    if (arr.length <2) {
        return null;                    T.C => O(n)
                                        S.C => O(1)
    }
    int largest = Integer.MIN_VALUE;
    int secondLargest = Integer.MIN_VALUE;

    for (int i=0; i<arr.length; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] >secondLargest && arr[i] != largest)
          { secondLargest = arr[i];
          }
    }
    return SecondLargest;
}
```

) Check if array is Sorted or not.

```
Public static boolean isSorted (int [] arr) {
    if (arr.length<1) {
        return true;
    }
    for(int i=1; i<arr.length; i++) {
        if (arr[i] < arr[i-1]) {
            return false;
        }
    }
    return true;
}
```

T.C => O(n)
S.C => O(1)

5) Remove Duplicates from Sorted Array

```
Public int removeDuplicates (int[] nums) {
    int n= nums.length;
    int i=0;
    for(int j=1; j<n; j++) {
        if (nums[i] != nums[j]) {
            i++;
            nums[i] = nums[j];
        }
    }
    return i+1;
}
```

T.C => O(n)
S.C => O(1)

6) Rotate a given array

```
Public static void rotateArray(int[] arr, int k) {
  if(arr==null || arr.length==0) {
    return;
  }
  if(K==0)
    return;
  reverse(arr,0, arr.length-1);
  reverse(arr, 0, K-1);
  reverse(arr, K, arr.length-1);

Public static void reverse(int[]arr, int start, int end){
  while(start<end){
    int temp= arr[start];
    arr[start]= arr[end];
    arr[end] = temp;
    start ++;
    end --;
  }
}
```

T.C => O(n)
S.C => O(1)

7) Find frequency of elements in a given array

```
Public Static Map<Integer, Integer> findfreq (int [] arr) {
  Map<Integer, Integer> map = new HashMap<>();

  for (int num : arr) {
    ~~freq~~
    map.put(num, map.getOrDefault(num,0)+1);
  }                                          T.C => O(n)
                                             S.C => O(n)
  return map;

}
```

8) Merge two sorted arrays

```
public static int[] merge(int[] nums1, int m, int[] nums2,
                          int n) {

  int i = m-1;
  int j = n-1;                               T.C => O(m+n)
  int k = m+n-1;                             SC => O(1)
  while (i>=0 && j>==0){
    if(nums1 [i] > nums[j]){
      nums1[k] = nums1[i];
      i--;
    }else{
      nums1[k] = nums2[j];
      j--;
    }
    k--;
  }
  while(j>=0){
    nums1[k] = nums2[j];
    j--;
    k--;
  } return nums1;
}
```