Assignment No 9

Que1 .Answer

```
public class TwoSum {

    public int[] twoSum(int[] numbers, int target) {
        int left = 0;
        int right = numbers.length - 1;

        while (left < right) {
            int sum = numbers[left] + numbers[right];

            if (sum == target) {
                return new int[]{left + 1, right + 1}; // Adding 1 to convert from 0-indexed to

            } else if (sum < target) {
                left++;
            } else {
                right--;
            }
        }

        // There is always one solution, so this return statement won't be executed.
        return new int[]{-1, -1};
    }

    public static void main(String[] args) {
        TwoSum twoSumII = new TwoSum();

        int[] numbers = {2, 7, 11, 15};
        int target = 9;
        int[] result = twoSumII.twoSum(numbers, target);
        System.out.println("Output: [" + result[0] + ", " + result[1] + "]");
    }
}
```

Que2 .Answer

```
public class FindTargetRange {
    public static int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};

        // Find the starting position
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] >= target) {
                right = mid - 1;
            }
        }
        if (left < nums.length && nums[left] == target) {
            result[0] = left;
        }

        // Find the ending position
        left = 0;
        right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] <= target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid - 1;
            }
        }
        if (right >= 0 && nums[right] == target) {
            result[1] = right;
        }

        return result;
    }

    public static void main(String[] args) {
        int[] nums = {5, 7, 7, 8, 8, 10};
        int target = 8;
        int[] output = searchRange(nums, target);
        System.out.println("Output: [" + output[0] + ", " + output[1] + "]");
    }
}
```

Que3 .Answer

```
public class FindPeakElement {
    public static int findPeakElement(int[] nums) {
        int left = 0, right = nums.length - 1;

        while (left < right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] < nums[mid + 1]) {
                // Move towards the higher neighbor
                left = mid + 1;
            } else {
                // Move towards the left side as we are interested in a peak element
                right = mid;
            }
        }

        return left;
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 3, 1};
        int peakIndex = findPeakElement(nums);
        System.out.println("Output: " + peakIndex); // Output: 2
    }
}
```

Que 4 .Answer

```
public class SearchInsertPosition {
    public static int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return left;

        // the  point, left represent the index where the target should be inserted
    }

    public static void main(String[] args) {
        int[] nums = {1, 3, 5, 6};
        int target1 = 5;
        int target2 = 7;

        int index1 = searchInsert(nums, target1);
        int index2 = searchInsert(nums, target2);

        System.out.println("Output  target 5: " + index1);
      System.out.println("Output  target 7: " + index2);
    }
}
```

Que 5. Answer

```
public class MajorityElement {
    public static int findMajorityElement(int[] nums) {
        int majorityElement = nums[0];
        int count = 1;

        for (int i = 1; i < nums.length; i++) {
            if (count == 0) {
                majorityElement = nums[i];
                count = 1;
            } else if (nums[i] == majorityElement) {
                count++;
            } else {
                count--;
            }
        }

        return majorityElement;
    }

    public static void main(String[] args) {
        int[] nums = {3, 3, 4, 2, 4, 4, 2, 4, 4};
        int majorityElement = findMajorityElement(nums);
        System.out.println("Output: " + majorityElement); // Output: 4
    }
}
```

Que 6.Answer

```
public class FirstBadVersion {

    private boolean isBadVersion(int version) {

        return version >= 4;
    }

    public int firstBadVersion(int n) {
        int left = 1;
        int right = n;

        while (left < right) {
            int mid = left + (right - left) / 2;

            if (isBadVersion(mid)) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }

        // At this point, left represents the first bad version
        return left;
    }

    public static void main(String[] args) {
        FirstBadVersion fbv = new FirstBadVersion();
        int n = 5;
        int firstBadVersion = fbv.firstBadVersion(n);
        System.out.println("Output: " + firstBadVersion);
    }
}
```

Que.7 Answer:

```
public class CountInversions {
    public static int countInversions(int[] arr) {
        int n = arr.length;
        int inversions = 0;

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

        return inversions;
    }

    public static void main(String[] args) {
        int[] arr = {2, 4, 1, 3, 5};
        int inversions = countInversions(arr);
        System.out.println("Output: " + inversions); // Output: 3
    }
}
```

Que. 8 Answer:

```
public class CommonElements {
   public static void findCommonElements (int[] ar1, int[] ar2, int[] ar3) {
      int i = 0, j = 0, k = 0;

      while (i < ar1.length && j < ar2.length && k < ar3.length)
{
         if (ar1[i] == ar2[j] && ar2[j] == ar3[k])
        {
            System.out.print(ar1[i] + " ");
            i++;
            j++;
            k++;
         } else if (ar1[i] < ar2[j]) {
            i++;
         } else if (ar2[j] < ar3[k]) {
            j++;
         } else {
            k++;
         }
      }
   }

   public static void main(String[] args) {
      int[] ar1 = {1, 5, 10, 20, 40, 80};
      int[] ar2 = {6, 7, 20, 80, 100};
      int[] ar3 = {3, 4, 15, 20, 30, 70, 80, 120};

      System.out.print("Output: ");
      findCommonElements(ar1, ar2, ar3);
 }
}
```