Assignment No. 5

PAGE NO.
DATE

Que: 1. Answer:-

```
public class Array converter {

public static int [ ][ ] convertTo2DArray
                    ( int [] original, int m, int n)
 {
   int [ ][ ] result = new int [m][n];
   if (original.length != m * n) {
      return new  int [0][0];
   }
    int index = 0;
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {

         result[i][j] = original[index++];
      }
   }
   return result;
 }
 public static void main (String [] args) {
   int [] original = {1, 2, 3, 4};
     int m = 2;
     int n = 2;
int [][] result = convertTo2DArray (original, m, n);

for( int i = 0; i < result.length; i++) {

   for (int j = 0; j < result[0].length; j++) {

    System.out.print(result[i][j] + " ");
   }
   system.out.println();
  }
 }
}
```

Output

```
1   2

3   4
```

Que: 2: Answer:-

PAGE NO.
DATE

```
public class Staircase {
  public static int findCompleteRows (int n) {
  int completeRows = 0;
  int totalCoins = 0;
   int i = 1;
  while( totalCoins + i <= n) {
      totalCoins += i;
      completeRows++;
      i++;
   }
   return completeRows ;
 }
 public static void main ( String[] args) {
  int n = 5;
  int completeRows = findCompleteRows (n);
  System.out.println(completeRows);
 }
}
```

Que: 3:- Answer:-

```
 import java.util.Arrays ;

public class squareSorted Array {
  public static int[] sortedSquares (int [] nums)
   {
     int[] result = new int [num.length];

   int left = 0;
   int right = num.length - 1;
   int index = num.length - 1;
     while (left <= right) {
   int leftSquare = nums [left] * nums [Left];
   int ~~left~~ right square = nums [right] *
                                 nums [left];
if (leftSquare > rightSquare) {
  result[index] = leftSquare;
  left++;
} else {
  result[index] = rightSquare;
  right--;
}
  index--;
}
return result;
}
  public static void main (String[] args) {

  int[] nums = {-4,-1,0,3,10};
  int[] result = sortedSquares(nums);

  System.out.println(Arrays.toString
                              (result));
  }

}
```

Q.4. Answer:

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class DistinctIntegers {
public static List<List<Integer>> findDistinct
         Integers(int[] num1, int[] num2) {

  List<List<Integer>> answer = new
                      ArrayList<>();
  Set<Integer>> ~~[illegible]~~ set1 = new HashSet<>();
  Set<Integer> set2 = new HashSet<>();
```

```
for (int num: nums1) {
   set.add(num);
}
 for (int num: nums2) {

   set2.add(num);

 }

List<Integer> distinctInNums1 = new ArrayList
                              <>();
List<Integer> distinctInNums2 = new ArrayList
                              <>();
for (int num : set1) {
   if (!set2.contains(num)) {
      distinctInNums1.add(num);
   }

}
 for (int num : set2) {
  if (!set1.contains(num)) {
     distinctInNums2.add(num);
   }

}
answer.add(distinctInNums1);
answer.add(distinctInNums2);

 return answer;
}
 public static void main(String[] args) {
 int[] nums1 = {1, 2, 3};
 int[] nums2 = {2, 4, 6};

List<List<Integer>> answer = findDistinct-
           Integers(nums1, nums2);
 System.out.println(answer);
 }
}
```

Que5:- Answer:-

```
public class DistanceValue {
public static int findDistanceValue(int[] arr1,
                        int[] arr2, int d) {
 int distance = 0;

 for (int num1 : arr1) {
   boolean isValid = true;

  for (int num2 : arr2) {
   if (Math.abs(num1 - num2) <= d) {
       isValid = false;
        break;
      }
   }
 if (isValid) {
      distance ++;
 }
 }
 return distance;
 }
  public static void main(String [] args) {

int [] arr1 = {4, 5, 8};
int [] arr2 = {10, 9, 1, 8};
int d = 2;
int distance = findDistanceValue(arr1, arr2, d);

System.out.println(distance);
 }
}
```

Que 6. Answer:-

```
 import java.util.ArrayList;
 import java.util.List;
public class FindDuplicates {
 public static List<Integer> findDuplicates(int[]
                                          nums) {
   List<Integer> duplicates = new ArrayList
                                          <>();
   for (int i = 0; i < nums.length; i++) {
     int index = Math.abs(nums[i]) - 1;

    if (nums[index] < 0) {
          duplicates.add(index + 1);
    } else {
    } nums[index] = -nums[index];

     }
  }
  return duplicates;
 }
 public static void main(String[] args) {

   int [] nums = {4, 3, 2, 7, 8, 2, 3, 1};
  List<Integer> duplicates = findDuplicates(nums);

  System.out.println(duplicates);
 }
}
```

Que 7. Answer:-

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

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

if (nums[mid] > nums [right]) {

   left = mid+1;
 } else {
   right = mid;
 }
}
return nums [left];
}
public static void main (String [] args) {

  int [] nums = {3, 4, 5, 1, 2};
  int min = findMin (nums);
  ~~Int~~
  System.out.println(min);
 }
}
```

Que 8. Answer:.

```
 import java.util.ArrayList;
 import java.util.*;
 import java.util.List;
 import java.util.HashMap;

public class DoubledArray {
 public static int[] findOriginalArray (int[]
                                        changed) {
   if (changed.length % 2 != 0) {
      return new int [0];
   }
  int [] original = new int[changed.length/2];
  Map< Integer, Integer> countMap = new
                             HashMap<>();
  for (int num: changed) {

     countMap. put(num, countMap.getOrDefault
                             (num,0) + 1);
  }
  Arrays. sort(changed);
  int index = 0;
 for (int num: changed) {
   if (countMap. get (num) > 0) {
    int doubledNum = num * 2;

if (countMap.containsKey(doubledNum) && countMap.
                 get (doubleNum) > 0).
   original [index] = num;
   countMap. put (num, countMap. get(num)-1);
  countMap.put(doubledNum, countMap. get
                         (doubleNum) -1);
  index++;
 }
```

PAGE NO.

DATE:

```
else {
    return new int[0]
  }
 }
}
 return original;
}
  public static void main (String[] args) {
   int[] changed = {1, 3, 4, 2, 6, 8};
   int [] ~~changed~~.
          original = findOriginalArray(changed);
   ~~System.out.println (Arrays.toString(original));~~
  }
}
```