NAME: ANEEQ SHAMS ROLL NO: BSE-230 SECTION: E

Lab #04

"Data Structures and Algorithms"

Exercises

Lab Tasks:

1. Write a program that takes two arrays of size 4 and swaps the elements of those arrays.

- Input:

```
import java.util.Arrays;
public class JavaApplication4Aneeq230 {
    public static void main(String[] args) {

       int Aneeq1[ ] = {5,  22, 35, 230};
        int Aneeq2[ ] = {100, 6, 98, 540};
        int temp[ ] = new int [4];

        System.out.println("BEFORE SWAPPING");
        System.out.println("---------------");
        System.out.println("Array 1: " + Arrays.toString(Aneeq1));
        System.out.println("Array 2: " + Arrays.toString(Aneeq2));

        for (int i=0; i<Aneeq1.length; i++) {
           temp[ i ] = Aneeq1 [ i ];
           Aneeq1[ i ] = Aneeq2[ i ];
           Aneeq2[ i ] = temp[ i ];
        }

        System.out.println(" ");
        System.out.println("AFTER SWAPPING");
        System.out.println("---------------");
        System.out.println("Array 1: " + Arrays.toString(Aneeq1));
        System.out.println("Array 2: " + Arrays.toString(Aneeq2));
    }
}
```

Output:

```
run:
BEFORE SWAPPING
---------------
Array 1: [5, 22, 35, 230]
Array 2: [100, 6, 98, 540]

AFTER SWAPPING
---------------
Array 1: [100, 6, 98, 540]
Array 2: [5, 22, 35, 230]
```

2. Add a method in the class that takes array and merge it with the existing one.

- Input:

```
import java.util.Arrays;

public class JavaApplication4Aneeq230 {
   public static int [ ] mergedArrays(int [ ] aneeq1, int [ ] aneeq2) {

      int length = aneeq1.length + aneeq2.length;
      int [ ] merged = new int [length];

      System.arraycopy(aneeq1, 0, merged, 0, aneeq1.length);
      System.arraycopy(aneeq2 ,0, merged, aneeq1.length, aneeq2.length);
      return merged;
   }

public static void main(String[] args) {
   int [ ] aneeq1= { 5, 7, 9, 10, 55};
   int [ ] aneeq2= {99, 72, 15, 17, 230};
   int [ ] mergedArray= mergedArrays(aneeq1, aneeq2);

  System.out.println("Array 1: " + Arrays.toString(aneeq1));
  System.out.println("Array 2: " + Arrays.toString(aneeq2));

 System.out.println ("Merged Array: " + Arrays.toString(mergedArray));
}
}
```

Output:

```
run:
Array 1: [5, 7, 9, 10, 55]
Array 2: [99, 72, 15, 17, 230]
Merged Array: [5, 7, 9, 10, 55, 99, 72, 15, 17, 230]
```

3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not.

- Input:

```
public class JavaApplication4Aneeq230 {
   public static void main(String[] args) {

      String[ ] S = {"rotator", "mom", "how", "peep", "why", "civic"};

      for (String word : S) {
         if (isPalindrome(word)) {
            System.out.println(word + " is a Palindrome.");
         } else {
            System.out.println(word + " is not a Palindrome.");
   }}
   }
public static boolean isPalindrome(String str) {
int a=0;
int b= str.length()-1;
while (a<b) {
 if (str.charAt(a) != str.charAt(b)) {
            return false;
         }
 a++;
 b--;
}
return true;
}
}
```

Output:

```
run:
rotator is a palindrome.
mom is a palindrome.
how is not a palindrome.
peep is a palindrome.
why is not a palindrome.
civic is a palindrome.
```

4. Given an array of integers, count how many numbers are even and how many are odd.

- Input:

```
public class JavaApplication4Aneeq230 {
    public static void main(String[] args) {

int [ ] num = {2, 4, 5, 12, 111, 15, 121};
int evenCount = 0;
int oddCount = 0;

for(int n : num) {
if(n%2==0)
  evenCount++;
else
  oddCount++;
}

System.out.println("Even Numbers are:" + evenCount);
System.out.println("Odd Numbers are:" + oddCount);
}
}
```

Output:

```
run:
Even Numbers:3
Odd Numbers:4
```

5. Given two integer arrays, merge them and remove any duplicate values from the resulting array.

- Input:

```
public class JavaApplication4Aneeq230 {
  public static void main(String[ ] args) {
    int [ ] aneeq1 ={1, 5, 0, 230, 12, 5};
    int  [ ] aneeq2 ={5, 3, 6, 80, 1, 7};
    merged(aneeq1,aneeq2);
  }
  public static void merged(int[ ] aneeq1,int[ ] aneeq2 ){
      int temp []=new int [aneeq1.length+aneeq2.length];

      for (int i = 0; i < aneeq1.length; i++) {
          temp[i] = aneeq1[i];
      }
```

```
        for (int i = 0; i < aneeq2.length; i++) {
            temp[aneeq1.length + i] = aneeq2[i];
        }

        Set<Integer> uniqueElements = new LinkedHashSet<>();
        for (int e : temp) {
            uniqueElements.add(e);
        }
        System.out.print("The Merged Array is : ");
        for (int e : uniqueElements) {
            System.out.print(e + " ");
        }System.out.print("]");
        System.out.println();
    }
}
```

Output:

```
run:
The Merged Array is : 1 5 0 230 12 3 6 80 7 ]
```

Home Tasks:

1. Write a program that takes an array of Real numbers having size 7 and calculate the sum and mean of all the elements. Also depict the memory management of this task.

- Input:

```
import java.util.Scanner;
public class JavaApplication4Aneeq230 {
    public static void main(String[] args) {
        int [ ] array = Array(7);
        sumAndMean(array);
    }
    public static int [ ] Array(int n) {
        Scanner input = new Scanner(System.in);
        int no;
        int [ ] arr = new int[ n ];
        System.out.println("Enter Array Elements: ");
        for (int i=0; i<n; i++){
            no= input.nextInt();
            arr[ i ]= no;
        }
        return arr;
    }
    public static void sumAndMean(int[ ] arr) {
        int sum=0;
        int count=0;
        for (int a:arr) {
            sum+=a;
            count++;
        }
        System.out.println("The sum of an Array is: " + sum);
        System.out.println("The Mean of an Array is: " + (sum/count));
        }
        }
```

Output:

```
run:
Enter Array Elements:
2
2
4
4
6
6
2
The sum of an Array is: 26
The Mean of an Array is: 3
```

2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key

- Input;

```
public class JavaApplication4Aneeq230 {
    public static int[ ][ ] splitArray(int[ ] arr, int key) {
        int index = -1;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                index = i;
                break;
            }
        }
        if (index != -1) {
            int[ ] array1 = new int[index + 1];
            int[ ] array2 = new int[arr.length - index - 1];
            for (int i = 0; i <= index; i++) {
                array1[i] = arr[i];
            }
            for (int i = index + 1; i < arr.length; i++) {
                array2[i - index - 1] = arr[i];
            }
            return new int[ ][ ] { array1, array2};
        } else {
            return new int[ ][ ] { arr };
        }
    }
    public static void main(String[] args) {
        int[ ] arr = {230, 22, 76, 4, 9, 110};
        int key = 76;
        int[ ][ ] result = splitArray(arr, key);
        System.out.println("First Array will be: ");
        for (int i : result[0]) {
            System.out.print(i + " ");
        }
        System.out.println();
        System.out.println("Array's key will be: 76" );
        System.out.println();
        if (result.length > 1) {
            System.out.println("Second Array will be: ");
            for (int i : result[1]) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output:

```
run:
First Array will be:
230 22 76
Array's key will be: 76

Second Array will be:
4 9 110 BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Given an array of distinct integers and a target integer, return all unique combinations of numbers that add up to the target. Each number can be used only once in the combination.

- Input:

```
import java.util.Arrays;
public class JavaApplication4Aneeq230 {
  public static void main(String[] args){
   int[ ] n = {4, 2, 1, 3, 5, 7, 6};
      int target = 8;

      System.out.println("Combinations that sum to " + target + ":");
      findCombinations(n, target);
   }
   public static void findCombinations(int[ ] n, int target) {
      Arrays.sort(n);
      int [ ] combination = new int [n.length];
      findCombinations(n, target, 0, combination, 0);
   }
   private static void findCombinations(int[ ] n, int target, int start, int[ ] combination, int pos) {
      if (target == 0) {
         for (int i = 0; i < pos; i++) {
            System.out.print(combination[i] + " ");
         }
         System.out.println();
         return;
      }
       for (int i = start; i < n.length; i++) {
          if (n [ i ] > target) break;
          combination [ pos ] = n [ i ];
          findCombinations(n, target - n [ i ], i + 1, combination, pos + 1);
       }
   }
}
```

Output:

```
run:
Combinations that sum to 8:
1 2 5
1 3 4
1 7
2 6
3 5
```

4. You are given an array containing n distinct numbers taken from 0, 1, 2, ..., n. Write a program to find the one number that is missing from the array.

- Input:

```
public class JavaApplication4Aneeq230 {
public static void main(String[] args){
    int [ ] aneeq = {5, 4, 2, 0, 1};
     int missed = findM(aneeq);
     System.out.println("The missing number is: " + missed);
  }
  public static int findM(int [ ] no) {
     int n = no.length;
     int expSum = n * (n + 1) / 2;
     int actSum = 0;
     for (int e : no) {
        actSum += e;
     }
     return expSum - actSum;
  }
}
```

Output:

```
run:
The missing number is: 3
```

5. You are given an array of integers. Write a program to sort the array such that it follows a zigzag pattern: the first element is less than the second, the second is greater than the third, and so on.

- Input:

```
public class JavaApplication4Aneeq230 {
public static void main(String[] args){
    int [ ] aneeq = {0, 8, 2, 7, 230, 5, 10000, 10};
     zSort(aneeq);
     System.out.println("Zigzag Pattern Array is: ");
     for (int num : aneeq) {
        System.out.print(num + " ");
     }
  }
  public static void zSort(int [ ] array) {
     for (int i = 0; i < array.length - 1; i++) {
        if (i % 2 == 0) {
           if (array [ i ] > array [i + 1]) {
              int store = array [ i ];
              array [ i ] = array [i + 1];
              array [i + 1] =store;
           }
        } else {
           if (array [ i ] < array [i + 1]) {
              int store = array [ i ];
              array [ i ] = array [i + 1];
              array [i + 1] = store;
           }
        }
     }
  }}
```

Output:

```
run:
Zigzag Pattern Array is:
0 8 2 230 5 10000 7 10 BUILD SUCCESSFUL (total time: 0 seconds)
```