LAB # 05

Sorting on Linear Array

OBJECTIVE: To sort a linear array using Selection Sort, Bubble Sort and Merge Sort.

LAB TASKS

1. Write a program for Selection sort that sorts an array containing numbers, prints all the sort values of array each followed by its location.

CODE :

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {64, 25, 12, 22, 11};
        for (int i = 0; i < arr.length - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[j] < arr[minIdx]) {
                    minIdx = j;
                }
            }
            int temp = arr[minIdx];
            arr[minIdx] = arr[i];
            arr[i] = temp;
            System.out.println(arr[i] + " " + i);
        }
    }
}
```

OUTPUT:

```
11 0
12 1
22 2
25 3
```

1. Write a program that takes 10 numbers as input in an array. Sort the elements of array by using Bubble sort. Print each iteration of the sorting process.

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {5, 1, 4, 2, 8};
        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
            for (int num : arr) System.out.print(num + " ");
            System.out.println();
        }
    }
}
```

Output:

```
1 4 2 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
```

2. Write a program that takes 10 random numbers in an array. Sort the elements of array by using Merge sort applying recursive technique. Print each iteration of the sorting process.

Code:

```
public class Main {
    public static void mergeSort(int[] arr, int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    public static void merge(int[] arr, int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;
        int[] L = new int[n1];
        int[] R = new int[n2];
        for (int i = 0; i < n1; ++i) L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j) R[j] = arr[m + 1 + j];
        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
        while (i < n1) arr[k++] = L[i++];
        while (j < n2) arr[k++] = R[j++];
    }
    public static void main(String[] args) {
        int[] arr = {12, 11, 13, 5, 6, 7};
        mergeSort(arr, 0, arr.length - 1);
        for (int num : arr) System.out.print(num + " ");
    }
}
```

Output:

```
5 6 7 11 12 13
```

Home Tasks

1. Declare an array of size n to store account balances. Initialize with values 0 to 100000 and sort Account No's according to highest balance values by using Quick sort.

Code:

```java
public class Main {
    public static void quickSort(int[] balances, int[] accounts, int low, int high) {
        if (low < high) {
            int pi = partition(balances, accounts, low, high);
            quickSort(balances, accounts, low, pi - 1);
            quickSort(balances, accounts, pi + 1, high);
        }
    }
    public static int partition(int[] balances, int[] accounts, int low, int high) {
        int pivot = balances[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (balances[j] >= pivot) {
                i++;
                int temp = balances[i];
                balances[i] = balances[j];
                balances[j] = temp;

                temp = accounts[i];
                accounts[i] = accounts[j];
                accounts[j] = temp;
            }
        }
        int temp = balances[i + 1];
        balances[i + 1] = balances[high];
        balances[high] = temp;
        temp = accounts[i + 1];
        accounts[i + 1] = accounts[high];
        accounts[high] = temp;
        return i + 1;
    }
    public static void main(String[] args) {
        int[] balances = {28000, 12000, 45000, 8000, 31000};
        int[] accounts = {3547, 1245, 8791, 2354, 7482};
        quickSort(balances, accounts, 0, balances.length - 1);
        for (int i = 0; i < balances.length; i++) {
            System.out.println(accounts[i] + " " + balances[i]);
        }
    }
}
```

Output:

```
8791 45000
7482 31000
3547 28000
1245 12000
2354 8000
```

3. Write a program which takes an unordered list of integers (or any other objects e.g. String), you have to rearrange the list in their natural order using merge sort.

Code:

```
public class Main {
    public static void mergeSort(int[] arr, int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    public static void merge(int[] arr, int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;
        int[] L = new int[n1];
        int[] R = new int[n2];
        for (int i = 0; i < n1; i++) L[i] = arr[l + i];
        for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];
        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
        while (i < n1) arr[k++] = L[i++];
        while (j < n2) arr[k++] = R[j++];
    }
    public static void main(String[] args) {
        int[] arr = {28, 12, 15, 7, 34};
        mergeSort(arr, 0, arr.length - 1);
        for (int num : arr) System.out.println(num);
    }
}
```

Output:

```
7
12
15
28
34
```

2. You are given an unordered list of integers or strings. Write a program to Take this list as input. Sort it in natural order using Merge Sort. For integers, this means ascending order. For strings, this means alphabetical order. Print the sorted list.

Code:

```
import java.util.Scanner;
public class Main {
    public static void mergeSort(String[] arr, int l, int r) {
        if (l < r) {
            int m = (l + r) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    public static void merge(String[] arr, int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;
        String[] L = new String[n1];
        String[] R = new String[n2];

        for (int i = 0; i < n1; i++) L[i] = arr[l + i];
        for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];

        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) arr[k++] = (L[i].compareTo(R[j]) <= 0) ? L[i++] : R[j++];
        while (i < n1) arr[k++] = L[i++];
        while (j < n2) arr[k++] = R[j++];
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array:");
        int n = sc.nextInt();
        sc.nextLine();
        String[] arr = new String[n];
        System.out.println("Enter elements (integers or strings):");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextLine();
        }
        mergeSort(arr, 0, arr.length - 1);

        for (String s : arr) {
            System.out.println(s);
        }
    }
}
```

Output:

```
Enter size of array:
5
Enter elements (integers or strings):
a
b
2
a
1
1
2
a
a
b
```

4. You are given a set of bank accounts, each with a unique account number and a balance. Write a Java program to Declare an array of size n to store account balances. Initialize each balance randomly with values between 0 and 100,000. Sort the accounts in descending order of their balances using Quick Sort. Print the sorted list in the format.

Code:

```java
import java.util.Random;
public class Main {
    public static void quickSort(int[] balances, int[] accounts, int low, int high) {
        if (low < high) {
            int pi = partition(balances, accounts, low, high);
            quickSort(balances, accounts, low, pi - 1);
            quickSort(balances, accounts, pi + 1, high);
        }
    }
    public static int partition(int[] balances, int[] accounts, int low, int high) {
        int pivot = balances[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (balances[j] >= pivot) {
                i++;
                int temp = balances[i];
                balances[i] = balances[j];
                balances[j] = temp;

                temp = accounts[i];
                accounts[i] = accounts[j];
                accounts[j] = temp;
            }
        }
        int temp = balances[i + 1];
        balances[i + 1] = balances[high];
        balances[high] = temp;
        temp = accounts[i + 1];
        accounts[i + 1] = accounts[high];
        accounts[high] = temp;
        return i + 1;
    }
    public static void main(String[] args) {
        int n = 10;
        int[] balances = new int[n];
        int[] accounts = new int[n];
        Random rand = new Random();
        for (int i = 0; i < n; i++) {
            balances[i] = rand.nextInt(100001);
            accounts[i] = 1000 + i;
        }
        quickSort(balances, accounts, 0, n - 1);
        for (int i = 0; i < n; i++) {
            System.out.println(accounts[i] + " " + balances[i]);
        }
    }
}
```

Output:

```
1000 99177
1007 88609
1004 82990
1005 82329
1006 72763
1008 64281
1001 62600
1003 51638
1009 27634
1002 14349
```