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.

Source Code:

```
import java.util.Scanner;
public class Lab5Task1 {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      System.out.print("Enter length of array: ");
      int[] array = new int[scan.nextInt()];
      for (int i = 0; i < array.length; i++) {
         System.out.print("Enter element at position '" + i + "': ");
         array[i] = scan.nextInt();}
      selectionSort(array);
      scan.close();}
   public static void selectionSort(int[] array) {
      for (int i = 0; i < array.length - 1; i++) {
         int low = i;
         for (int j = i + 1; j < array.length; j++) {
            if (array[j] < array[low]) low = j;}
         int temp = array[i]; array[i] = array[low]; array[low] = temp;}
      for (int element : array) System.out.print(element + ", ");
      System.out.println();}}
```

Output:

```
Output - Lab5Task1 (run)
run:
Enter length of array: 5
Enter element at position '0': 77
Enter element at position '1': 55
Enter element at position '2': 22
Enter element at position '3': 88
Enter element at position '4': 33
22, 33, 55, 77, 88,
BUILD SUCCESSFUL (total time: 11 seconds)
```

2. 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.

Source Code:

```
import java.util.Scanner;
public class Lab5Task2 {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      System.out.print("Enter length of array: ");
      int[] array = new int[scan.nextInt()];
      for (int i = 0; i < array.length; i++) {
         System.out.print("Enter element at position '" + i + "': ");
         array[i] = scan.nextInt();}
      bubbleSort(array);
      scan.close();}
   public static void bubbleSort(int[] array) {
      int n = array.length;
      for (int i = 0; i < n; i++) {
         for (int j = 1; j < n - i; j++) {
            if (array[j - 1] > array[j]) {
               int temp = array[j - 1];
               array[j - 1] = array[j];
               array[j] = temp;}}
         System.out.print("array[] = ");
         for (int k : array) System.out.print(k + ", ");
         System.out.println();}}}
```

Output:

```
Output - Lab5Task2 (run)
run:
Enter length of array: 10
Enter element at position '0': 99
Enter element at position '1': 77
Enter element at position '2': 88
Enter element at position '3': 55
Enter element at position '4': 66
Enter element at position '5': 44
Enter element at position '6': 33
Enter element at position '7': 22
Enter element at position '8': 11
Enter element at position '9': 111
array[] = 77, 88, 55, 66, 44, 33, 22, 11, 99, 111,
array[] = 77, 55, 66, 44, 33, 22, 11, 88, 99, 111,
array[] = 55, 66, 44, 33, 22, 11, 77, 88, 99, 111,
array[] = 55, 44, 33, 22, 11, 66, 77, 88, 99, 111,
array[] = 44, 33, 22, 11, 55, 66, 77, 88, 99, 111,
array[] = 33, 22, 11, 44, 55, 66, 77, 88, 99, 111,
array[] = 22, 11, 33, 44, 55, 66, 77, 88, 99, 111,
array[] = 11, 22, 33, 44, 55, 66, 77, 88, 99, 111,
array[] = 11, 22, 33, 44, 55, 66, 77, 88, 99, 111,
array[] = 11, 22, 33, 44, 55, 66, 77, 88, 99, 111,
BUILD SUCCESSFUL (total time: 24 seconds)
```

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

Source Code:

```
class Lab5Task3 {
   void merge(int arr[], int l, int m, int r) {
      int n1 = m - l + 1, n2 = r - m;
      int[] L = new int[n1], R = new int[n2];
      System.arraycopy(arr, l, L, 0, n1);
      System.arraycopy(arr, m + 1, R, 0, n2);
      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++];}
   void sort(int arr[], int l, int r) {
      if (l < r) {
         int m = l + (r - l) / 2;
         sort(arr, l, m);
         sort(arr, m + 1, r);
         merge(arr, l, m, r);}}
   static void printArray(int arr[]) {
      for (int num : arr) System.out.print(num + " ");
      System.out.println();}
   public static void main(String[] args) {
      int arr[] = {12, 11, 13, 5, 6, 7};
      System.out.println("Given Array");
      printArray(arr);
      Lab5Task3 ob = new Lab5Task3();
      ob.sort(arr, 0, arr.length - 1);
      System.out.println("\nSorted Array");
      printArray(arr);}}
```

Output:

```
Output - Lab5Task3 (run)
run:
Given Array
12 11 13 5 6 7

Sorted Array
5 6 7 11 12 13
BUILD SUCCESSFUL (total time: 0 seconds)
```

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, For e.g.:

- Account No. 3547 Balance 28000
- Account No. 1245 Balance 12000

Source Code:

```
import java.util.*;
public class Lab5HomeTask1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the size of an array: ");
      int size = sc.nextInt();
      int[] arr = new int[size];
      for (int i = 0; i < size; arr[i++] = sc.nextInt()) {
         System.out.print("Enter the balance: ");}
      quickSort(arr, 0, arr.length - 1);
      for (int i = arr.length - 1; i >= 0; i--) {
         System.out.printf("Account No %d Balance: %d%n", i + 1, arr[i]);}}
   private static void quickSort(int[] arr, int low, int high) {
      if (low < high) {
         int pi = partition(arr, low, high);
         quickSort(arr, low, pi - 1);
         quickSort(arr, pi + 1, high);}}
   private static int partition(int[] arr, int low, int high) {
      int pivot = arr[high], i = low - 1;
      for (int j = low; j < high; j++) {
         if (arr[j] < pivot) {
            swap(arr, ++i, j);}}
      swap(arr, i + 1, high);
      return i + 1;}
   private static void swap(int[] arr, int i, int j) {
      int temp = arr[i];
      arr[i] = arr[j];
      arr[j]=temp;}}
```

Output:

```
Output - Lab5HomeTask1 (run)
run:
Enter the size of an array: 5
Enter the balance: 500
Enter the balance: 100
Enter the balance: 1000
Enter the balance: 2500
Enter the balance: 8000
Account No 5 Balance: 8000
Account No 4 Balance: 2500
Account No 3 Balance: 1000
Account No 2 Balance: 500
Account No 1 Balance: 100
BUILD SUCCESSFUL (total time: 19 seconds)
```

2. 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.

Source Code:

```
import java.util.*;
public class Lab5HomeTask2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of an array: ");
        int size = sc.nextInt();
        int[] arr = new int[size];
        System.out.print("Enter Elements of array: ");
        for (int i = 0; i < size; arr[i++] = sc.nextInt());
        System.out.println("Array before sorting: " + Arrays.toString(arr));
        mergesort(arr, 0, arr.length - 1);
        System.out.println("Array after sorting using mergesort algorithm: " + Arrays.toString(arr));}
    private static void mergesort(int[] arr, int start, int end) {
        if (start < end) {
            int mid = (start + end) / 2;
            mergesort(arr, start, mid);
            mergesort(arr, mid + 1, end);
            merge(arr, start, mid, end);}}
    private static void merge(int[] arr, int start, int mid, int end) {
        int[] tmp = new int[end - start + 1];
        int i = 0, left = start, right = mid + 1;
        while (left <= mid && right <= end) {
            tmp[i++] = arr[left] < arr[right] ? arr[left++] : arr[right++];}
        while (left <= mid) tmp[i++] = arr[left++];
        while (right <= end) tmp[i++] = arr[right++];
        System.arraycopy(tmp, 0, arr, start, tmp.length);}}
```

Output:

```
Output - Lab5HomeTask2 (run)
run:
Enter the size of an array: 8
Enter Elements of array: 22
95
13
16
145
288
477
252
Array before sorting: [22, 95, 13, 16, 145, 288, 477, 252]
Array after sorting using mergesort algorithm: [13, 16, 22, 95, 145, 252, 288, 477]
BUILD SUCCESSFUL (total time: 35 seconds)
```

3. 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.

Source Code:

```
import java.util.*;
public class Lab5HomeTask3{
   static void mergeSort(int[] arr, int left, int right) {
      if (left < right) {
         int mid = (left + right) / 2;
         mergeSort(arr, left, mid);
         mergeSort(arr, mid + 1, right);
         merge(arr, left, mid, right);}}
   static void merge(int[] arr, int left, int mid, int right) {
      int[] temp = new int[right - left + 1];
      int i = left, j = mid + 1, k = 0;
      while (i <= mid && j <= right) temp[k++] = arr[i] <= arr[j] ? arr[i++] : arr[j++];
      while (i <= mid) temp[k++] = arr[i++];
      while (j <= right) temp[k++] = arr[j++];
      System.arraycopy(temp, 0, arr, left, temp.length);}
   static void mergeSort(String[] arr, int left, int right) {
      if (left < right) {
         int mid = (left + right) / 2;
         mergeSort(arr, left, mid);
         mergeSort(arr, mid + 1, right);
         merge(arr, left, mid, right);}}
   static void merge(String[] arr, int left, int mid, int right) {
      String[] temp = new String[right - left + 1];
      int i = left, j = mid + 1, k = 0;
      while (i <= mid && j <= right) temp[k++] = arr[i].compareTo(arr[j]) <= 0 ? arr[i++] : arr[j++];
      while (i <= mid) temp[k++] = arr[i++];
      while (j <= right) temp[k++] = arr[j++];
      System.arraycopy(temp, 0, arr, left, temp.length);}
   public static void main(String[] args) {
      int[] intArray = {55, 66, 44, 33, 22, 11, 77, 88, 99};
      mergeSort(intArray, 0, intArray.length - 1);
      System.out.println("Sorted integers: " + Arrays.toString(intArray));
      String[] stringArray = {"My Name is Khurram Raza","I'm CR of Section A Batch 2023F 3rd Semester"};
      mergeSort(stringArray, 0, stringArray.length - 1);
      System.out.println("Sorted strings: " + Arrays.toString(stringArray));}}
```

Output:

```
Output - Lab5HomeTask3 (run)
run:
Sorted integers: [11, 22, 33, 44, 55, 66, 77, 88, 99]
Sorted strings: [I'm CR of Section A Batch 2023F 3rd Semester, My Name is Khurram Raza]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Source Code:

```
import java.util.*;
class BankAccount {
   int accountNumber, balance;
   BankAccount(int acc, int bal) {
      accountNumber = acc;
      balance = bal;}
   @Override
   public String toString() {
      return "Account: " + accountNumber + ", Balance: " + balance;}}
public class Lab5HomeTask4 {
   static void quickSort(BankAccount[] arr, int low, int high) {
      if (low < high) {
         int pivot = partition(arr, low, high);
         quickSort(arr, low, pivot - 1);
         quickSort(arr, pivot + 1, high);}}
   static int partition(BankAccount[] arr, int low, int high) {
      BankAccount pivot = arr[high];
      int i = low - 1;
      for (int j = low; j < high; j++) {
         if (arr[j].balance > pivot.balance) swap(arr, ++i, j);}
      swap(arr, i + 1, high);
      return i + 1;}
   static void swap(BankAccount[] arr, int i, int j) {
      BankAccount temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;}
   public static void main(String[] args) {
      Random rand = new Random();
      BankAccount[] accounts = new BankAccount[5];
      for (int i = 0; i < 5; i++) accounts[i] = new BankAccount(i + 1, rand.nextInt(100001));
      System.out.println("Before sorting:");
      Arrays.stream(accounts).forEach(System.out::println);
      quickSort(accounts, 0, accounts.length - 1);
      System.out.println("\nAfter sorting:");
      Arrays.stream(accounts).forEach(System.out::println);}}
```

Output:

```
Output - Lab5HomeTask4 (run)
run:
Before sorting:
Account: 1, Balance: 43105
Account: 2, Balance: 37219
Account: 3, Balance: 95271
Account: 4, Balance: 80771
Account: 5, Balance: 92474

After sorting:
Account: 3, Balance: 95271
Account: 5, Balance: 92474
Account: 4, Balance: 80771
Account: 1, Balance: 43105
Account: 2, Balance: 37219
BUILD SUCCESSFUL (total time: 0 seconds)
```