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

DATA STRUCTURES AND ALGORITHMS

LAB NO # 05

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.**

INPUT :

```
public class SelectionSort {
    public static void main(String[] args) {
        int hussainArray[] = {70, 54, 119, 31, 23}; // Array to be sorted

        // Selection sort
        for (int i = 0; i < hussainArray.length - 1; i++) {
            int smallest = i; // Assume the smallest element is at index 'i'

            // Find the smallest element in the unsorted part of the array
            for (int j = i + 1; j < hussainArray.length; j++) {
                if (hussainArray[j] < hussainArray[smallest]) {
                    smallest = j; // Update the smallest element index
                }
            }

            // Swap the smallest element with the first unsorted element
            int temp = hussainArray[smallest];
            hussainArray[smallest] = hussainArray[i];
            hussainArray[i] = temp;

            // Print the current sorted value along with its location (index)
            System.out.println("Sorted Value: " + hussainArray[i] + " at location: " + i);
        }

        // Print the final sorted array
        System.out.print("Final Sorted Array: ");
        for (int i = 0; i < hussainArray.length; i++) {
            System.out.print(hussainArray[i] + " ");
        }
        System.out.println(); // Print a new line after the array
    }
}
```

Output ;

```
run:
Sorted Value: 23 at location: 0
Sorted Value: 31 at location: 1
Sorted Value: 54 at location: 2
Sorted Value: 70 at location: 3
Final Sorted Array: 23 31 54 70 119
BUILD SUCCESSFUL (total time: 0 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.**

Input :

```
import java.util.Scanner;

public class BubbleSortExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create an array named arrHussain to store 10 numbers
        int[] arrHussain = new int[10];

        // Take 10 numbers as input from the user
        System.out.println("Enter 10 numbers:");
        for (int i = 0; i < 10; i++) {
            arrHussain[i] = scanner.nextInt();
        }

        // Perform Bubble Sort
        for (int i = 0; i < arrHussain.length - 1; i++) {
            // Flag to check if any swapping happened in this iteration
            boolean swapped = false;

            // Loop through the array from 0 to n-i-1
            for (int j = 0; j < arrHussain.length - 1 - i; j++) {
                // If the current element is greater than the next element, swap them
                if (arrHussain[j] > arrHussain[j + 1]) {
                    // Swap the elements
                    int temp = arrHussain[j];
                    arrHussain[j] = arrHussain[j + 1];
                    arrHussain[j + 1] = temp;

                    swapped = true; // Mark that a swap has occurred
                }
            }


            // Print the final sorted array
            System.out.print("Sorted Array: ");
            printArray(arrHussain);
        }

        // Method to print the array
        public static void printArray(int[] arrHussain) {
            for (int num : arrHussain) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
}
```

Output :

```
run:
Enter 10 numbers:
80
0
60
50
0
30
20
10
9
5
Iteration 1: 0 60 50 0 30 20 10 9 5 80
Iteration 2: 0 50 0 30 20 10 9 5 60 80
Iteration 3: 0 0 30 20 10 9 5 50 60 80
Iteration 4: 0 0 20 10 9 5 30 50 60 80
Iteration 5: 0 0 10 9 5 20 30 50 60 80
Iteration 6: 0 0 9 5 10 20 30 50 60 80
Iteration 7: 0 0 5 9 10 20 30 50 60 80
Iteration 8: 0 0 5 9 10 20 30 50 60 80
Sorted Array: 0 0 5 9 10 20 30 50 60 80
BUILD SUCCESSFUL (total time: 25 seconds)
```

3. **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.**

Input :

```java
import java.util.Random;

public class MergeSortExample {

    public static void main(String[] args) {
        // Create an array named arrHussain with 10 random numbers
        int[] arrHussain = new int[10];
        Random random = new Random();

        // Fill the array with random numbers
        System.out.println("Original Array:");
        for (int i = 0; i < arrHussain.length; i++) {
            arrHussain[i] = random.nextInt(100); // Random numbers between 0 and 99
            System.out.print(arrHussain[i] + " ");
        }
        System.out.println("\n");

        // Perform Merge Sort with recursive technique
        mergeSort(arrHussain, 0, arrHussain.length - 1);
    }

    // Merge Sort recursive method
    public static void mergeSort(int[] arrHussain, int left, int right) {
        if (left < right) {
            // Find the middle point
            int mid = (left + right) / 2;

            // Recursively sort the first and second halves
            mergeSort(arrHussain, left, mid);
            mergeSort(arrHussain, mid + 1, right);
```

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

```
            // Merge the sorted halves
            merge(arrHussain, left, mid, right);

            // Print the array after each merge step
            System.out.print("Array after merge: ");
            printArray(arrHussain);
        }
    }

    // Merge method to combine the sorted subarrays
    public static void merge(int[] arrHussain, int left, int mid, int right) {
        // Find the size of two subarrays to be merged
        int n1 = mid - left + 1;
        int n2 = right - mid;

        // Temporary arrays to hold the values
        int[] leftArray = new int[n1];
        int[] rightArray = new int[n2];

        // Copy the data to the temporary arrays
        System.arraycopy(arrHussain, left, leftArray, 0, n1);
        System.arraycopy(arrHussain, mid + 1, rightArray, 0, n2);

        // Merge the temporary arrays back into the original array
        int i = 0, j = 0, k = left;
        while (i < n1 && j < n2) {
            if (leftArray[i] <= rightArray[j]) {
                arrHussain[k] = leftArray[i];
                i++;
            } else {
                arrHussain[k] = rightArray[j];
                j++;
            }
            k++;
        }

        // Copy the remaining elements of leftArray, if any
        while (i < n1) {
            arrHussain[k] = leftArray[i];
            i++;
            k++;
        }

        // Copy the remaining elements of rightArray, if any
        while (j < n2) {
            arrHussain[k] = rightArray[j];
            j++;
            k++;
        }
    }

    // Method to print the array
    public static void printArray(int[] arrHussain) {
        for (int num : arrHussain) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output :

```
run:
Original Array:
27 61 3 17 62 49 6 43 52 18

Array after merge: 27 61 3 17 62 49 6 43 52 18
Array after merge: 3 27 61 17 62 49 6 43 52 18
Array after merge: 3 27 61 17 62 49 6 43 52 18
Array after merge: 3 17 27 61 62 49 6 43 52 18
Array after merge: 3 17 27 61 62 6 49 43 52 18
Array after merge: 3 17 27 61 62 6 43 49 52 18
Array after merge: 3 17 27 61 62 6 43 49 18 52
Array after merge: 3 17 27 61 62 6 18 43 49 52
Array after merge: 3 6 17 18 27 43 49 52 61 62
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

INPUT :

```java
import java.util.Random;

public class UniqueQuickSortAccounts {

    public static void main(String[] args) {
        int totalAccounts = 10; // Number of accounts
        int[] accIDs = new int[totalAccounts]; // Array to store account IDs
        int[] accBalances = new int[totalAccounts]; // Array to store account balances

        // Initialize account IDs and balances
        Random random = new Random();
        System.out.println("Initial Account Details:");
        for (int i = 0; i < totalAccounts; i++) {
            accIDs[i] = 1000 + random.nextInt(9000); // Generate random 4-digit account IDs
            accBalances[i] = random.nextInt(100001); // Generate random balances between 0 and 100,000
            System.out.println("Account ID: " + accIDs[i] + ", Balance: " + accBalances[i]);
        }

        // Sort account balances in descending order using Quick Sort
        sortAccounts(accBalances, accIDs, 0, totalAccounts - 1);

        // Display sorted account details
        System.out.println("\nSorted Account Details (By Balance in Descending Order):");
        for (int i = 0; i < totalAccounts; i++) {
            System.out.println("Account ID: " + accIDs[i] + ", Balance: " + accBalances[i]);
        }
    }

    // Sort method (Quick Sort logic)
    public static void sortAccounts(int[] balances, int[] ids, int start, int end) {
        if (start < end) {
            // Partition the array and get the pivot index
            int pivotIndex = partition(balances, ids, start, end);
```

```
            int pivotIndex = partition(balances, ids, start, end);

            // Sort the two halves
            sortAccounts(balances, ids, start, pivotIndex - 1);
            sortAccounts(balances, ids, pivotIndex + 1, end);
        }
    }

    // Partition method for Quick Sort
    public static int partition(int[] balances, int[] ids, int start, int end) {
        int pivotValue = balances[end]; // Choose the last element as pivot
        int i = start - 1; // Index of the smaller element

        for (int j = start; j < end; j++) {
            if (balances[j] > pivotValue) { // For descending order
                i++;
                // Swap balances[i] and balances[j]
                swap(balances, i, j);
                // Swap corresponding account IDs
                swap(ids, i, j);
            }
        }

        // Place pivot element in the correct position
        swap(balances, i + 1, end);
        swap(ids, i + 1, end);

        return i + 1; // Return the index of the pivot
    }

    // Swap method to simplify swapping logic
    public static void swap(int[] array, int a, int b) {
        int temp = array[a];
        array[a] = array[b];
        array[b] = temp;
    }
}
```

Output :

```
run:
Initial Account Details:
Account ID: 8075, Balance: 26414
Account ID: 9119, Balance: 82756
Account ID: 3757, Balance: 73051
Account ID: 1533, Balance: 47986
Account ID: 5238, Balance: 47442
Account ID: 6388, Balance: 75224
Account ID: 9003, Balance: 11313
Account ID: 8953, Balance: 2117
Account ID: 1459, Balance: 55253
Account ID: 6827, Balance: 65256

Sorted Account Details (By Balance in Descending Order):
Account ID: 9119, Balance: 82756
Account ID: 6388, Balance: 75224
Account ID: 3757, Balance: 73051
Account ID: 6827, Balance: 65256
Account ID: 1459, Balance: 55253
Account ID: 1533, Balance: 47986
Account ID: 5238, Balance: 47442
Account ID: 8075, Balance: 26414
Account ID: 9003, Balance: 11313
Account ID: 8953, Balance: 2117
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Home tasks 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.

INPUT :

```
import java.util.Arrays;
import java.util.Scanner;

public class MergeSortListHussain {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input: Take an unordered list of integers
        System.out.print("Enter the number of elements in the list: ");
        int n = scanner.nextInt();
        Integer[] arrHussain = new Integer[n]; // Array named arrHussain for uniqueness

        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            arrHussain[i] = scanner.nextInt();
        }

        // Display original list
        System.out.println("Unordered List (arrHussain): " + Arrays.toString(arrHussain));

        // Sort the list using Merge Sort
        mergeSort(arrHussain, 0, arrHussain.length - 1);

        // Display sorted list
        System.out.println("List in Natural Order (arrHussain): " + Arrays.toString(arrHussain));
    }

    // Merge Sort function
    public static void mergeSort(Integer[] arrHussain, int left, int right) {
        if (left < right) {
            // Find the middle point
            int mid = left + (right - left) / 2;

            // Recursively sort the two halves
            mergeSort(arrHussain, left, mid);
            mergeSort(arrHussain, mid + 1, right);

            // Merge the sorted halves
            merge(arrHussain, left, mid, right);
        }
    }

    // Merge function to combine two sorted halves
    public static void merge(Integer[] arrHussain, int left, int mid, int right) {
        // Sizes of the two subarrays
        int n1 = mid - left + 1;
        int n2 = right - mid;

        // Temporary arrays
        Integer[] leftArray = new Integer[n1];
        Integer[] rightArray = new Integer[n2];

        // Copy data to temporary arrays
        for (int i = 0; i < n1; i++) {
            leftArray[i] = arrHussain[left + i];
        }
        for (int j = 0; j < n2; j++) {
            rightArray[j] = arrHussain[mid + 1 + j];
        }

        // Merge the temporary arrays back into the original array
        int i = 0, j = 0;
        int k = left;
        while (i < n1 && j < n2) {
```

```
            if (leftArray[i] <= rightArray[j]) {
                arrHussain[k] = leftArray[i];
                i++;
            } else {
                arrHussain[k] = rightArray[j];
                j++;
            }
            k++;
        }

        // Copy any remaining elements of the left array
        while (i < n1) {
            arrHussain[k] = leftArray[i];
            i++;
            k++;
        }

        // Copy any remaining elements of the right array
        while (j < n2) {
            arrHussain[k] = rightArray[j];
            j++;
            k++;
        }
    }
```

Output :

```
run:
Enter the number of elements in the list: 5
Enter 5 integers:
20
0
12
88
67
Unordered List (arrHussain): [20, 0, 12, 88, 67]
List in Natural Order (arrHussain): [0, 12, 20, 67, 88]
BUILD SUCCESSFUL (total time: 22 seconds)
```

Home tasks 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.

INPUT :

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

```java
import java.util.Arrays;
import java.util.Scanner;

public class MergeSortHussain {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // User chooses the type of list: integers or strings
        System.out.println("Choose the type of list:");
        System.out.println("1. Integer List");
        System.out.println("2. String List");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        if (choice == 1) {
            // Integer List
            System.out.print("Enter the number of integers: ");
            int n = scanner.nextInt();
            Integer[] arrHussain = new Integer[n]; // Unique array name

            System.out.println("Enter " + n + " integers:");
            for (int i = 0; i < n; i++) {
                arrHussain[i] = scanner.nextInt();
            }

            // Display and sort
            System.out.println("Original Integer List (arrHussain): " + Arrays.toString(arrHussain));
            mergeSort(arrHussain, 0, arrHussain.length - 1);
            System.out.println("Sorted Integer List (arrHussain): " + Arrays.toString(arrHussain));

        } else if (choice == 2) {
            // String List
            System.out.print("Enter the number of strings: ");
            int n = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            String[] arrHussain = new String[n]; // Unique array name

            System.out.println("Enter " + n + " strings:");
            for (int i = 0; i < n; i++) {
                arrHussain[i] = scanner.nextLine();
            }

            // Display and sort
            System.out.println("Original String List (arrHussain): " + Arrays.toString(arrHussain));
            mergeSort(arrHussain, 0, arrHussain.length - 1);
            System.out.println("Sorted String List (arrHussain): " + Arrays.toString(arrHussain));
        } else {
            System.out.println("Invalid choice! Please run the program again.");
        }
    }

    // Generic Merge Sort function
    public static <T extends Comparable<T>> void mergeSort(T[] arrHussain, int left, int right) {
        if (left < right) {
            int mid = left + (right - left) / 2;

            // Recursively sort the two halves
            mergeSort(arrHussain, left, mid);
            mergeSort(arrHussain, mid + 1, right);

            // Merge the sorted halves
            merge(arrHussain, left, mid, right);
        }
    }
```

Output :

```
run:
Choose the type of list:
1. Integer List
2. String List
1
Enter the number of integers: 5
Enter 5 integers:
100
250
187
34
78
Original Integer List (arrHussain): [100, 250, 187, 34, 78]
Sorted Integer List (arrHussain): [34, 78, 100, 187, 250]
BUILD SUCCESSFUL (total time: 29 seconds)
```

Home tasks 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

INPUT :

```
import java.util.Arrays;
import java.util.Random;

public class BankAccountsHussain {
    public static void main(String[] args) {
        int n = 10; // Number of accounts
        int[][] arrHussain = new int[n][2]; // Array to store account numbers and balances

        Random random = new Random();

        // Initialize account numbers and random balances
        for (int i = 0; i < n; i++) {
            arrHussain[i][0] = 1000 + i; // Account numbers start from 1000
            arrHussain[i][1] = random.nextInt(100001); // Random balance between 0 and 100,000
        }

        // Display unsorted accounts
        System.out.println("Unsorted Accounts (arrHussain):");
        printAccounts(arrHussain);

        // Sort accounts in descending order by balance using Quick Sort
        quickSort(arrHussain, 0, n - 1);

        // Display sorted accounts
        System.out.println("\nSorted Accounts by Balance (arrHussain):");
        printAccounts(arrHussain);
    }

    // Quick Sort method
    public static void quickSort(int[][] arrHussain, int low, int high) {
        if (low < high) {
            int pi = partition(arrHussain, low, high); // Partition index
            quickSort(arrHussain, low, pi - 1); // Recursively sort left subarray
```

```
            quickSort(arrHussain, pi + 1, high); // Recursively sort right subarray
        }
    }

    // Partition method for Quick Sort
    public static int partition(int[][] arrHussain, int low, int high) {
        int pivot = arrHussain[high][1]; // Use last element as pivot (balance)
        int i = low - 1; // Index of smaller element

        for (int j = low; j < high; j++) {
            // If current balance is greater than pivot, swap
            if (arrHussain[j][1] > pivot) {
                i++;
                swap(arrHussain, i, j);
            }
        }

        // Swap the pivot element with the element at index i + 1
        swap(arrHussain, i + 1, high);
        return i + 1;
    }

    // Swap method to exchange rows in the array
    public static void swap(int[][] arrHussain, int i, int j) {
        int[] temp = arrHussain[i];
        arrHussain[i] = arrHussain[j];
        arrHussain[j] = temp;
    }

    // Method to print account details
    public static void printAccounts(int[][] arrHussain) {
        System.out.printf("%-15s%-15s\n", "Account No.", "Balance");
          for (int[] account : arrHussain) {
              System.out.printf("%-15d%-15d\n", account[0], account[1]);
          }
     }
}
```

Output :

```
run:
Unsorted Accounts (arrHussain):
Account No.     Balance
1000            38019
1001            56396
1002            19970
1003            54584
1004            59736
1005            97212
1006            35143
1007            55691
1008            92134
1009            33053

Sorted Accounts by Balance (arrHussain):
Account No.     Balance
1005            97212
1008            92134
1004            59736
1001            56396
1007            55691
1003            54584
1000            38019
1006            35143
1009            33053
1002            19970
BUILD SUCCESSFUL (total time: 0 seconds)
```