DataStructures and algorithms

Lab 7

Name:Pranjal bajaj

Cms:023-21-0022

Task 1

Bubble Sort

public class Sorts {

    public static void Sorts(int[] arr) {

        int n = arr.length;

        int comparisons = 0;

        int swaps = 0;

        for (int i = 0; i < n - 1; i++) {

            for (int j = 0; j < n - i - 1; j++) {

                comparisons++;

                if (arr[j] > arr[j + 1]) {

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                    swaps++;

                }

            }

        }

        System.out.println("Bubble Sort - Comparisons: " + comparisons + ", Swaps: " + swaps);

    }

    public static void main(String[] args) {

        int[] descOrder = {9, 7, 5, 3, 2};

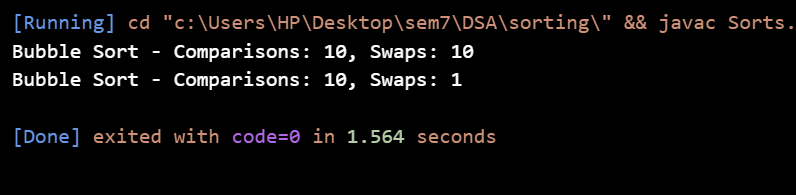
        int[] almostSorted = {1, 2, 3, 5, 4};

        Sorts(descOrder.clone());

        Sorts(almostSorted.clone());

    }

}



Selection Sort

public class Sorts {

    public static void Sorts(int[] arr) {

        int n = arr.length;

        int comparisons = 0;

        int swaps = 0;

        for (int i = 0; i < n - 1; i++) {

            int minIdx = i;

            for (int j = i + 1; j < n; j++) {

                comparisons++;

                if (arr[j] < arr[minIdx]) {

                    minIdx = j;

                }

            }

            if (minIdx != i) {

                int temp = arr[minIdx];

                arr[minIdx] = arr[i];

                arr[i] = temp;

                swaps++;

            }

        }

        System.out.println("Selection Sort - Comparisons: " + comparisons + ", Swaps: " + swaps);

    }

    public static void main(String[] args) {

        int[] descOrder = {9, 7, 5, 3, 2};

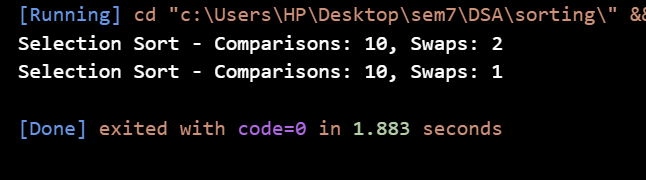
        int[] almostSorted = {1, 2, 3, 5, 4};

        Sorts(descOrder.clone());

        Sorts(almostSorted.clone());

    }

}



Insertion Sort

public class Sorts {

    public static void Sorts(int[] arr) {

        int n = arr.length;

        int comparisons = 0;

        int swaps = 0;

        for (int i = 1; i < n; i++) {

            int key = arr[i];

            int j = i - 1;

            while (j >= 0 && arr[j] > key) {

                comparisons++;

                arr[j + 1] = arr[j];

                j--;

                swaps++;

            }

            arr[j + 1] = key;

            if (j >= 0) comparisons++;

        }

        System.out.println("Insertion Sort - Comparisons: " + comparisons + ", Swaps: " + swaps);

    }

    public static void main(String[] args) {

        int[] descOrder = {9, 7, 5, 3, 2};

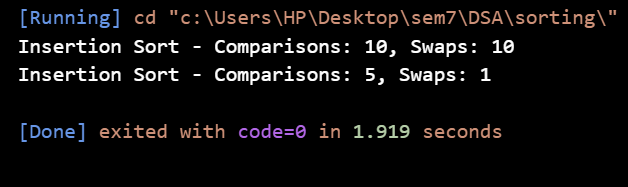
        int[] almostSorted = {1, 2, 3, 5, 4};

        Sorts(descOrder.clone());

        Sorts(almostSorted.clone());

    }

}



Task 2

import java.util.Arrays;

public class Sorts {

    public static void main(String[] args) {

        int[] array1 = {12, 7, 19, 3};

        int[] array2 = {4, 17, 5, 11};

        int[] mergedArray = new int[array1.length + array2.length];

        System.arraycopy(array1, 0, mergedArray, 0, array1.length);

        System.arraycopy(array2, 0, mergedArray, array1.length, array2.length);

        Arrays.sort(mergedArray);

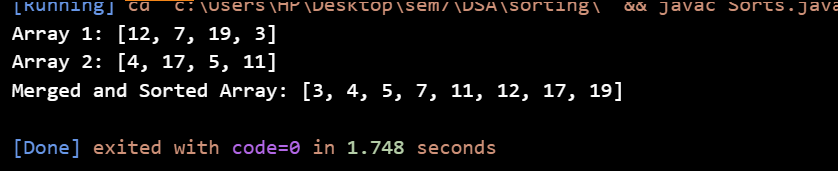
        System.out.println("Array 1: " + Arrays.toString(array1));

        System.out.println("Array 2: " + Arrays.toString(array2));

        System.out.println("Merged and Sorted Array: " + Arrays.toString(mergedArray));

    }

}



Task 3

import java.util.Arrays;

import java.util.Comparator;

class Student {

    String name;

    String cmsId;

    double gpa;

    public Student(String name, String cmsId, double gpa) {

        this.name = name;

        this.cmsId = cmsId;

        this.gpa = gpa;

    }

    public double getGpa() {

        return gpa;

    }

    @Override

    public String toString() {

        return "Name: " + name + ", CMS-ID: " + cmsId + ", GPA: " + gpa;

    }

}

public class Sorts {

    public static void main(String[] args) {

        Student[] students = {

            new Student("John Doe", "CMS123", 3.5),

            new Student("Jane Smith", "CMS456", 3.8),

            new Student("Alice Johnson", "CMS789", 3.2),

            new Student("Bob Brown", "CMS101", 3.9)

        };

        System.out.println("Before Sorting:");

        for (Student s : students) {

            System.out.println(s);

        }

        Arrays.sort(students, Comparator.comparingDouble(Student::getGpa).reversed());

        System.out.println("\nAfter Sorting by GPA (Descending):");

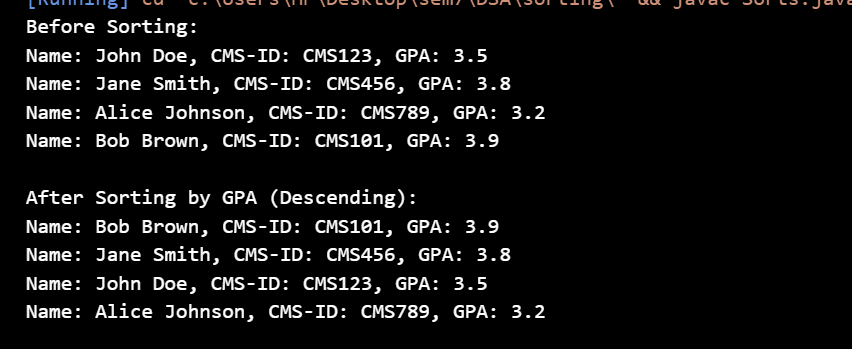
        for (Student s : students) {

            System.out.println(s);

        }

    }

}



Task 4

class Node {

    int data;

    Node next;

    public Node(int data) {

        this.data = data;

        this.next = null;

    }

}

class LinkedList {

    Node head;

    public void add(int data) {

        Node newNode = new Node(data);

        if (head == null) {

            head = newNode;

        } else {

            Node current = head;

            while (current.next != null) {

                current = current.next;

            }

            current.next = newNode;

        }

    }

    public void printList() {

        Node current = head;

        while (current != null) {

            System.out.print(current.data + " ");

            current = current.next;

        }

        System.out.println();

    }

    public void insertionSort() {

        if (head == null || head.next == null) {

            return;

        }

        Node sorted = null;

        Node current = head;

        while (current != null) {

            Node next = current.next;

            sorted = sortedInsert(sorted, current);

            current = next;

        }

        head = sorted;

    }

    private Node sortedInsert(Node sorted, Node newNode) {

        if (sorted == null || sorted.data >= newNode.data) {

            newNode.next = sorted;

            return newNode;

        }

        Node current = sorted;

        while (current.next != null && current.next.data < newNode.data) {

            current = current.next;

        }

        newNode.next = current.next;

        current.next = newNode;

        return sorted;

    }

}

public class Sorts {

    public static void main(String[] args) {

        LinkedList list = new LinkedList();

        list.add(29);

        list.add(23);

        list.add(82);

        list.add(11);

        list.add(4);

        System.out.println("Before Sorting:");

        list.printList();

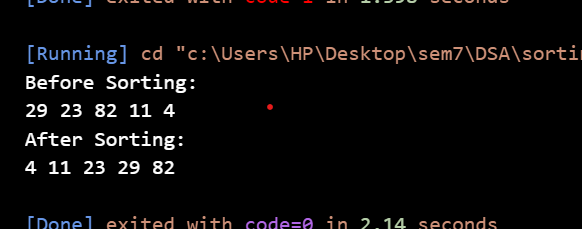
        list.insertionSort();

        System.out.println("After Sorting:");

        list.printList();

    }

}



Task 5

import java.util.Arrays;

import java.util.Comparator;

public class Sorts {

    public static String reconstructSentence(String s) {

        String[] words = s.split(" ");

        Arrays.sort(words, Comparator.comparingInt(word -> word.charAt(word.length() - 1) - '0'));

        StringBuilder originalSentence = new StringBuilder();

        for (String word : words) {

            originalSentence.append(word.substring(0, word.length() - 1)).append(" ");

        }

        return originalSentence.toString().trim();

    }

    public static void main(String[] args) {

        String shuffledSentence = "sentence4 a3 is2 This1";

        String shuffledSentence1 = "Myself2 Me1 I4 and3";

        System.out.println(shuffledSentence);

        String originalSentence = reconstructSentence(shuffledSentence);

        System.out.println(originalSentence);

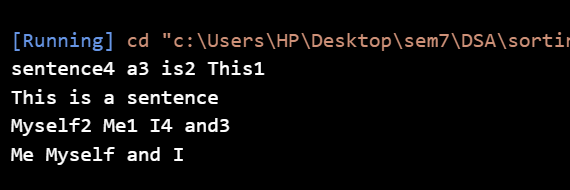
        System.out.println(shuffledSentence1);

        String originalSentence1 = reconstructSentence(shuffledSentence1);

        System.out.println(originalSentence1);

    }

}



Task 6

 class Sorts {

    public static void sortColors(int[] nums) {

        int low = 0, mid = 0, high = nums.length - 1;

        while (mid <= high) {

            switch (nums[mid]) {

                case 0: // Red (0)

                    swap(nums, low, mid);

                    low++;

                    mid++;

                    break;

                case 1: // White (1)

                    mid++;

                    break;

                case 2: // Blue (2)

                    swap(nums, mid, high);

                    high--;

                    break;

            }

        }

    }

    private static void swap(int[] nums, int i, int j) {

        int temp = nums[i];

        nums[i] = nums[j];

        nums[j] = temp;

    }

    public static void main(String[] args) {

        int[] nums = {2, 0, 2, 1, 1, 0};

        System.out.println("Before sorting:");

        for (int num : nums) {

            System.out.print(num + " ");

        }

        sortColors(nums);

        System.out.println("\nAfter sorting:");

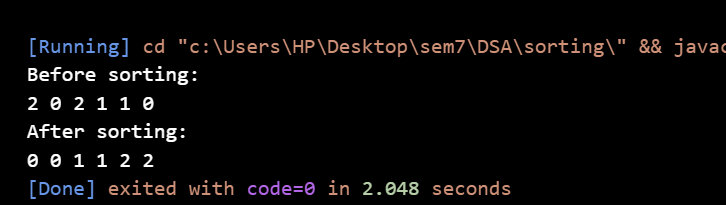
        for (int num : nums) {

            System.out.print(num + " ");

        }

    }

}



Task 7

class Sorts {

    public static void rearrangeArray(int[] nums) {

        int oddIndex = 1;

        int evenIndex = 0;

        while (oddIndex < nums.length && evenIndex < nums.length) {

            if (nums[evenIndex] % 2 == 0) {

                evenIndex += 2;

            } else if (nums[oddIndex] % 2 != 0) {

                oddIndex += 2;

            } else {

                swap(nums, evenIndex, oddIndex);

                evenIndex += 2;

                oddIndex += 2;

            }

        }

    }

    private static void swap(int[] nums, int i, int j) {

        int temp = nums[i];

        nums[i] = nums[j];

        nums[j] = temp;

    }

    public static void main(String[] args) {

        int[] nums = {32,  50, 47,65, 10, 83, 7, 2};

        System.out.println("Before rearrangement:");

        for (int num : nums) {

            System.out.print(num + " ");

        }

        rearrangeArray(nums);

        System.out.println("\nAfter rearrangement:");

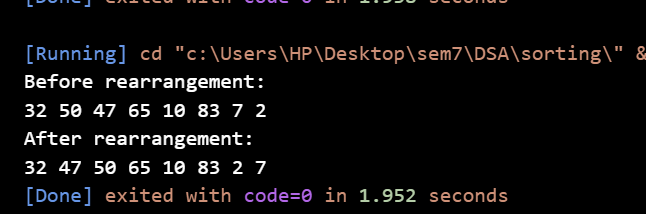
        for (int num : nums) {

            System.out.print(num + " ");

        }

    }

}



Task 8

import java.util.\*;

public class Sorts {

    public static String frequencySort(String s) {

        Map<Character, Integer> frequencyMap = new HashMap<>();

        for (char c : s.toCharArray()) {

            frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

        }

        List<Map.Entry<Character, Integer>> entryList = new ArrayList<>(frequencyMap.entrySet());

        entryList.sort((a, b) -> b.getValue() - a.getValue());

        StringBuilder sortedString = new StringBuilder();

        for (Map.Entry<Character, Integer> entry : entryList) {

            for (int i = 0; i < entry.getValue(); i++) {

                sortedString.append(entry.getKey());

            }

        }

        return sortedString.toString();

    }

    public static void main(String[] args) {

        String s = "tree";

        String result = frequencySort(s);

        System.out.println(result);

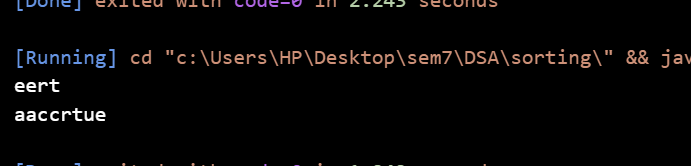
        String s1 = "accurate";

        String result1 = frequencySort(s1);

        System.out.println(result1);

    }

}



Task 9

class ListNode {

    int val;

    ListNode next;

    ListNode(int x) { val = x; }

}

public class Sorts {

    public static ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        ListNode dummy = new ListNode(0);

        ListNode current = dummy;

        while (list1 != null && list2 != null) {

            if (list1.val <= list2.val) {

                current.next = list1;

                list1 = list1.next;

            } else {

                current.next = list2;

                list2 = list2.next;

            }

            current = current.next;

        }

        if (list1 != null) {

            current.next = list1;

        } else {

            current.next = list2;

        }

        return dummy.next;

    }

    public static void printList(ListNode head) {

        ListNode current = head;

        while (current != null) {

            System.out.print(current.val + " ");

            current = current.next;

        }

        System.out.println();

    }

    public static void main(String[] args) {

        ListNode list1 = new ListNode(1);

        list1.next = new ListNode(2);

        list1.next.next = new ListNode(4);

        ListNode list2 = new ListNode(1);

        list2.next = new ListNode(3);

        list2.next.next = new ListNode(4);

        ListNode mergedList = mergeTwoLists(list1, list2);

        printList(mergedList);

    }

}

