Problem statement 1

Task-Level 1

Write the program to search the string in large document in java using IntelliJ IDE.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class StringSearch {

   public static boolean searchInFile(String filePath, String searchString) {
      try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
         String line;
         while ((line = br.readLine()) != null) {
            if (line.contains(searchString)) {
               return true;
            }
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
      return false;
   }

   public static void main(String[] args) {
      String filePath = "path/to/your/large/document.txt";
      String searchString = "text_to_search";

      boolean found = searchInFile(filePath, searchString);
      if (found) {
         System.out.println("String found in the document!");
      } else {
         System.out.println("String not found in the document.");
      }
   }
}
```

Task Level 2

Write the program to create string and based on the following string methods in java using IntelliJ IDE
a. to Uppercase.

b. to Lowercase

c. length

d. char At (int index)

e. substring

```
public class StringMethodsDemo {

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

```
        String str = "Welcome to Java Programming!";

        String upperCase = str.toUpperCase();
        System.out.println("Uppercase: " + upperCase);

        String lowerCase = str.toLowerCase();
        System.out.println("Lowercase: " + lowerCase);

        int length = str.length();
        System.out.println("Length of the string: " + length);

        char charAt = str.charAt(11);
        System.out.println("Character at index 11: " + charAt);

        String substring = str.substring(11, 15);
        System.out.println("Substring (11 to 15): " + substring);
    }
}
```

Task-Level 3

Write the program based on the following sort in java using IntelliJ IDE

a. Selection sort

b. Insertion sort

c. Bubble sort

```
public class SortingAlgorithms {

    public static void main(String[] args) {
        int[] arr1 = {64, 25, 12, 22, 11};
        int[] arr2 = {64, 25, 12, 22, 11};
        int[] arr3 = {64, 25, 12, 22, 11};

        selectionSort(arr1);
        insertionSort(arr2);
        bubbleSort(arr3);

        System.out.println("Selection Sort: ");
        printArray(arr1);

        System.out.println("Insertion Sort: ");
        printArray(arr2);

        System.out.println("Bubble Sort: ");
        printArray(arr3);
    }

    public static void selectionSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[minIdx]) {
```

```
                minIdx = j;
            }
        }
        int temp = arr[minIdx];
        arr[minIdx] = arr[i];
        arr[i] = temp;
    }
  }

  public static void insertionSort(int[] arr) {
    int n = arr.length;
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
  }

  public static void bubbleSort(int[] arr) {
    int n = arr.length;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
  }

  public static void printArray(int[] arr) {
    for (int i : arr) {
        System.out.print(i + " ");
    }
    System.out.println();
  }
}
```

Problem statement 2

Task-Level 1

Write the program to declare the array and initialize the array and write the program to track student grades using an Arrays in java using IntelliJ IDE.

```
import java.util.Arrays;

public class StudentGrades {

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

```
        // Declare and initialize the array
        String[] students = {"Alice", "Bob", "Charlie", "Diana"};
        int[] grades = {85, 92, 78, 90};

        // Display student grades
        System.out.println("Student Grades:");
        for (int i = 0; i < students.length; i++) {
            System.out.println(students[i] + ": " + grades[i]);
        }

        // Calculate and display the average grade
        double average = Arrays.stream(grades).average().orElse(0.0);
        System.out.println("Average Grade: " + average);

        // Find and display the highest and lowest grades
        int highestGrade = Arrays.stream(grades).max().orElse(0);
        int lowestGrade = Arrays.stream(grades).min().orElse(0);
        System.out.println("Highest Grade: " + highestGrade);
        System.out.println("Lowest Grade: " + lowestGrade);
    }
}
```

Task - Level 2

Write the program to search for duplicate in an array in java using IntelliJ IDE

```
import java.util.HashSet;
import java.util.Set;

public class FindDuplicates {

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5, 3, 6, 7, 8, 1};

        findDuplicates(array);
    }

    public static void findDuplicates(int[] array) {
        Set<Integer> seen = new HashSet<>();
        Set<Integer> duplicates = new HashSet<>();

        for (int num : array) {
            if (seen.contains(num)) {
                duplicates.add(num);
            } else {
                seen.add(num);
            }
```

```
        }

        if (duplicates.isEmpty()) {
            System.out.println("No duplicates found.");
        } else {
            System.out.println("Duplicates found: " + duplicates);
        }
    }
}
```

Task-Level 3

Write the program based on the following sort in java using IntelliJ IIDE

```
import java.util.Arrays;

public class SortingAlgorithms {

    public static void main(String[] args) {
        int[] array1 = {34, 7, 23, 32, 5, 62, 32, 32};
        int[] array2 = array1.clone();
        int[] array3 = array1.clone();

        System.out.println("Original array: " + Arrays.toString(array1));

        quickSort(array1, 0, array1.length - 1);
        System.out.println("Quick Sort: " + Arrays.toString(array1));

        mergeSort(array2, 0, array2.length - 1);
        System.out.println("Merge Sort: " + Arrays.toString(array2));

        shellSort(array3);
        System.out.println("Shell Sort: " + Arrays.toString(array3));
    }

    // Quick Sort
    public static void quickSort(int[] array, int low, int high) {
        if (low < high) {
            int pi = partition(array, low, high);
            quickSort(array, low, pi - 1);
            quickSort(array, pi + 1, high);
        }
    }

    private static int partition(int[] array, int low, int high) {
        int pivot = array[high];
        int i = (low - 1);
```

```
    for (int j = low; j < high; j++) {
        if (array[j] < pivot) {
            i++;
            int temp = array[i];
            array[i] = array[j];
            array[j] = temp;
        }
    }
    int temp = array[i + 1];
    array[i + 1] = array[high];
    array[high] = temp;
    return i + 1;
}

// Merge Sort
public static void mergeSort(int[] array, int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(array, left, mid);
        mergeSort(array, mid + 1, right);
        merge(array, left, mid, right);
    }
}

private static void merge(int[] array, int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int[] L = new int[n1];
    int[] R = new int[n2];

    System.arraycopy(array, left, L, 0, n1);
    System.arraycopy(array, mid + 1, R, 0, n2);

    int i = 0, j = 0;
    int k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            array[k] = L[i];
            i++;
        } else {
            array[k] = R[j];
            j++;
        }
        k++;
    }
```

```
        while (i < n1) {
            array[k] = L[i];
            i++;
            k++;
        }

        while (j < n2) {
            array[k] = R[j];
            j++;
            k++;
        }
    }

    // Shell Sort
    public static void shellSort(int[] array) {
        int n = array.length;
        for (int gap = n / 2; gap > 0; gap /= 2) {
            for (int i = gap; i < n; i++) {
                int temp = array[i];
                int j;
                for (j = i; j >= gap && array[j - gap] > temp; j -= gap) {
                    array[j] = array[j - gap];
                }
                array[j] = temp;
            }
        }
    }
}
```

Problem statement 3

Task-Level 1

Write the code to create indexOf function in java using IntelliJ IDE.

```
public class CustomIndexOf {

    public static void main(String[] args) {
        String str = "Welcome to Java Programming!";
        String subStr = "Java";

        int index = customIndexOf(str, subStr);
        System.out.println("Index of '" + subStr + "': " + index);
    }
```

```
    public static int customIndexOf(String str, String subStr) {
        int strLength = str.length();
        int subStrLength = subStr.length();

        for (int i = 0; i <= strLength - subStrLength; i++) {
            int j;
            for (j = 0; j < subStrLength; j++) {
                if (str.charAt(i + j) != subStr.charAt(j)) {
                    break;
                }
            }
            if (j == subStrLength) {
                return i;
            }
        }
        return -1;
    }
}
```

Task - Level 2

Write the program based on interface listIterator<E> and import java. collections and import java.ListIterator in java using IntelliJ IDE.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.ListIterator;

public class ListIteratorExample {

    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        Collections.addAll(list, "Java", "Python", "C++", "JavaScript");

        ListIterator<String> iterator = list.listIterator();

        while (iterator.hasNext()) {
            System.out.println("Next element: " + iterator.next());
        }

        System.out.println("Iterating in reverse:");

        while (iterator.hasPrevious()) {
            System.out.println("Previous element: " + iterator.previous());
        }
    }
}
```

Task - Level 3

Write the program based on doubly linked list in collections in java using IntelliJ IDE and create the linked list in java.

```
import java.util.LinkedList;
import java.util.ListIterator;

public class DoublyLinkedListExample {

   public static void main(String[] args) {
      LinkedList<String> doublyLinkedList = new LinkedList<>();

      doublyLinkedList.add("Node1");
      doublyLinkedList.add("Node2");
      doublyLinkedList.add("Node3");
      doublyLinkedList.add("Node4");

      System.out.println("Original List:");
      for (String node : doublyLinkedList) {
         System.out.println(node);
      }

      System.out.println("Traversing forward using ListIterator:");
      ListIterator<String> iterator = doublyLinkedList.listIterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      System.out.println("Traversing backward using ListIterator:");
      while (iterator.hasPrevious()) {
         System.out.println(iterator.previous());
      }

      doublyLinkedList.addFirst("NewNodeAtStart");
      doublyLinkedList.addLast("NewNodeAtEnd");

      System.out.println("List after adding elements at the start and end:");
      for (String node : doublyLinkedList) {
         System.out.println(node);
      }
   }
}
```

Problem statement 4

Task - Level 1

Write the program based on trees in java using IntelliJ IDE.

```
public class BinaryTreeExample {

   static class TreeNode {
      int value;
      TreeNode left, right;

      TreeNode(int item) {
         value = item;
         left = right = null;
      }
```

```
    }

    static class BinaryTree {
        TreeNode root;

        BinaryTree() {
            root = null;
        }

        void insert(int value) {
            root = insertRec(root, value);
        }

        TreeNode insertRec(TreeNode root, int value) {
            if (root == null) {
                root = new TreeNode(value);
                return root;
            }

            if (value < root.value) {
                root.left = insertRec(root.left, value);
            } else if (value > root.value) {
                root.right = insertRec(root.right, value);
            }

            return root;
        }

        void inorder() {
            inorderRec(root);
            System.out.println();
        }

        void inorderRec(TreeNode root) {
            if (root != null) {
                inorderRec(root.left);
                System.out.print(root.value + " ");
                inorderRec(root.right);
            }
        }

        public static void main(String[] args) {
            BinaryTree tree = new BinaryTree();

            tree.insert(50);
            tree.insert(30);
            tree.insert(20);
            tree.insert(40);
            tree.insert(70);
            tree.insert(60);
            tree.insert(80);

            System.out.println("In-order traversal of the binary tree:");
            tree.inorder();
        }
    }
}
```

Task - Level 2

Write the program to insert the nodes into trees in java using IntelliJ IDE.

```
public class BinaryTreeInsertion {

   static class TreeNode {
      int value;
      TreeNode left, right;

      TreeNode(int item) {
         value = item;
         left = right = null;
      }
   }

   static class BinaryTree {
      TreeNode root;

      BinaryTree() {
         root = null;
      }

      void insert(int value) {
         root = insertRec(root, value);
      }

      TreeNode insertRec(TreeNode root, int value) {
         if (root == null) {
            root = new TreeNode(value);
            return root;
         }

         if (value < root.value) {
            root.left = insertRec(root.left, value);
         } else if (value > root.value) {
            root.right = insertRec(root.right, value);
         }

         return root;
      }

      void inorder() {
         inorderRec(root);
         System.out.println();
      }

      void inorderRec(TreeNode root) {
         if (root != null) {
            inorderRec(root.left);
            System.out.print(root.value + " ");
            inorderRec(root.right);
         }
      }
```

```
    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        tree.insert(50);
        tree.insert(30);
        tree.insert(20);
        tree.insert(40);
        tree.insert(70);
        tree.insert(60);
        tree.insert(80);

        System.out.println("In-order traversal of the binary tree:");
        tree.inorder();
    }
  }
}
```

Task - Level 3

Write the program to insert the node algorithms in a tree and delete the node algorithms from a tree in java using IntelliJ IDE.

```
public class BinarySearchTree {

  static class TreeNode {
    int value;
    TreeNode left, right;

    TreeNode(int item) {
      value = item;
      left = right = null;
    }
  }

  static class BST {
    TreeNode root;

    BST() {
      root = null;
    }

    void insert(int value) {
      root = insertRec(root, value);
    }

    TreeNode insertRec(TreeNode root, int value) {
      if (root == null) {
        root = new TreeNode(value);
        return root;
      }

      if (value < root.value) {
        root.left = insertRec(root.left, value);
      } else if (value > root.value) {
        root.right = insertRec(root.right, value);
      }
```

```
    return root;
}

void delete(int value) {
    root = deleteRec(root, value);
}

TreeNode deleteRec(TreeNode root, int value) {
    if (root == null) {
        return root;
    }

    if (value < root.value) {
        root.left = deleteRec(root.left, value);
    } else if (value > root.value) {
        root.right = deleteRec(root.right, value);
    } else {
        if (root.left == null) {
            return root.right;
        } else if (root.right == null) {
            return root.left;
        }

        root.value = minValue(root.right);

        root.right = deleteRec(root.right, root.value);
    }

    return root;
}

int minValue(TreeNode root) {
    int minValue = root.value;
    while (root.left != null) {
        minValue = root.left.value;
        root = root.left;
    }
    return minValue;
}

void inorder() {
    inorderRec(root);
    System.out.println();
}

void inorderRec(TreeNode root) {
    if (root != null) {
        inorderRec(root.left);
        System.out.print(root.value + " ");
        inorderRec(root.right);
    }
}

public static void main(String[] args) {
    BST tree = new BST();
```

```
        tree.insert(50);
        tree.insert(30);
        tree.insert(20);
        tree.insert(40);
        tree.insert(70);
        tree.insert(60);
        tree.insert(80);

        System.out.println("In-order traversal of the BST:");
        tree.inorder();

        System.out.println("Deleting node 20:");
        tree.delete(20);
        tree.inorder();

        System.out.println("Deleting node 30:");
        tree.delete(30);
        tree.inorder();

        System.out.println("Deleting node 50:");
        tree.delete(50);
        tree.inorder();
    }
  }
}
```

Problem statement 5

Task - Level 1

Write the program to create HashMap in collections in java using IntelliJ IDE.

```
import java.util.HashMap;
import java.util.Map;

public class HashMapExample {

   public static void main(String[] args) {
      HashMap<String, Integer> map = new HashMap<>();

      map.put("Apple", 40);
      map.put("Banana", 30);
      map.put("Cherry", 50);
      map.put("Date", 20);

      System.out.println("HashMap contents:");
      for (Map.Entry<String, Integer> entry : map.entrySet()) {
         System.out.println(entry.getKey() + ": " + entry.getValue());
      }

      System.out.println("Value for key 'Banana': " + map.get("Banana"));

      map.remove("Date");
      System.out.println("HashMap after removing 'Date':");
```

```
        for (Map.Entry<String, Integer> entry : map.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }

        System.out.println("Size of the HashMap: " + map.size());
    }
}
```

Task - Level 2

Write the program to search the HashMap using the following methods in java using IntelliJ IDE

a. containsKey(object key)

b. containsValue(object key)

c. get (object key)

```
import java.util.HashMap;
import java.util.Map;

public class HashMapSearchExample {

    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();

        map.put("Apple", 40);
        map.put("Banana", 30);
        map.put("Cherry", 50);
        map.put("Date", 20);

        String keyToSearch = "Banana";
        Integer valueToSearch = 50;

        if (map.containsKey(keyToSearch)) {
            System.out.println("HashMap contains key '" + keyToSearch + "'.");
        } else {
            System.out.println("HashMap does not contain key '" + keyToSearch + "'.");
        }

        if (map.containsValue(valueToSearch)) {
            System.out.println("HashMap contains value '" + valueToSearch + "'.");
        } else {
            System.out.println("HashMap does not contain value '" + valueToSearch + "'.");
        }

        Integer retrievedValue = map.get(keyToSearch);
        if (retrievedValue != null) {
            System.out.println("Value for key '" + keyToSearch + "': " + retrievedValue);
        } else {
            System.out.println("Key '" + keyToSearch + "' not found in the HashMap.");
        }
    }
}
```

Task - Level 3

Write the program to create the unsorted HashMap and add unordered key value pair and create tree map and use entry set to obtain all key value pairs use sorted () to sort the value.

```
import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;

public class HashMapToTreeMapExample {

   public static void main(String[] args) {
      HashMap<String, Integer> hashMap = new HashMap<>();
      hashMap.put("Banana", 30);
      hashMap.put("Apple", 40);
      hashMap.put("Cherry", 50);
      hashMap.put("Date", 20);

      TreeMap<String, Integer> treeMap = new TreeMap<>(hashMap);

      System.out.println("Sorted entries by key:");
      for (Map.Entry<String, Integer> entry : treeMap.entrySet()) {
         System.out.println(entry.getKey() + ": " + entry.getValue());
      }
   }
}
```

Problem statement 6

Task - Level 2

Write the program for implementing graph in java.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Graph {

   private Map<String, List<String>> adjacencyList;

   public Graph() {
      adjacencyList = new HashMap<>();
   }

   // Add a vertex to the graph
   public void addVertex(String vertex) {
      adjacencyList.putIfAbsent(vertex, new ArrayList<>());
   }

   // Add an edge between two vertices
   public void addEdge(String vertex1, String vertex2) {
      adjacencyList.putIfAbsent(vertex1, new ArrayList<>());
      adjacencyList.putIfAbsent(vertex2, new ArrayList<>());
      adjacencyList.get(vertex1).add(vertex2);
```

```
    adjacencyList.get(vertex2).add(vertex1); // For undirected graph
}

// Remove an edge between two vertices
public void removeEdge(String vertex1, String vertex2) {
    List<String> edges1 = adjacencyList.get(vertex1);
    List<String> edges2 = adjacencyList.get(vertex2);
    if (edges1 != null) {
        edges1.remove(vertex2);
    }
    if (edges2 != null) {
        edges2.remove(vertex1);
    }
}

// Remove a vertex from the graph
public void removeVertex(String vertex) {
    adjacencyList.values().forEach(e -> e.remove(vertex));
    adjacencyList.remove(vertex);
}

// Print the adjacency list
public void printGraph() {
    for (Map.Entry<String, List<String>> entry : adjacencyList.entrySet()) {
        System.out.print(entry.getKey() + " -> ");
        for (String neighbor : entry.getValue()) {
            System.out.print(neighbor + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    Graph graph = new Graph();

    // Adding vertices
    graph.addVertex("A");
    graph.addVertex("B");
    graph.addVertex("C");
    graph.addVertex("D");

    // Adding edges
    graph.addEdge("A", "B");
    graph.addEdge("A", "C");
    graph.addEdge("B", "D");
    graph.addEdge("C", "D");

    // Print the graph
    System.out.println("Graph:");
    graph.printGraph();

    // Remove an edge
    graph.removeEdge("A", "C");
    System.out.println("\nGraph after removing edge A-C:");
    graph.printGraph();

    // Remove a vertex
```

```
        graph.removeVertex("D");
        System.out.println("\nGraph after removing vertex D:");
        graph.printGraph();
    }
}
```

Task - Level 2

Write the program based on breadth first search and depth first search in java using IntelliJ IDE.

```
import java.util.*;

public class GraphSearch {

    private Map<String, List<String>> adjacencyList;

    public GraphSearch() {
        adjacencyList = new HashMap<>();
    }

    public void addVertex(String vertex) {
        adjacencyList.putIfAbsent(vertex, new ArrayList<>());
    }

    public void addEdge(String vertex1, String vertex2) {
        adjacencyList.putIfAbsent(vertex1, new ArrayList<>());
        adjacencyList.putIfAbsent(vertex2, new ArrayList<>());
        adjacencyList.get(vertex1).add(vertex2);
        adjacencyList.get(vertex2).add(vertex1);
    }

    public void bfs(String startVertex) {
        Set<String> visited = new HashSet<>();
        Queue<String> queue = new LinkedList<>();
        queue.add(startVertex);
        visited.add(startVertex);

        System.out.println("Breadth-First Search starting from " + startVertex + ":");

        while (!queue.isEmpty()) {
            String vertex = queue.poll();
            System.out.print(vertex + " ");

            for (String neighbor : adjacencyList.get(vertex)) {
                if (!visited.contains(neighbor)) {
                    visited.add(neighbor);
                    queue.add(neighbor);
                }
            }
        }
        System.out.println();
    }

    public void dfs(String startVertex) {
        Set<String> visited = new HashSet<>();
        dfsHelper(startVertex, visited);
```

```
        System.out.println();
    }

    private void dfsHelper(String vertex, Set<String> visited) {
        if (visited.contains(vertex)) return;
        visited.add(vertex);
        System.out.print(vertex + " ");

        for (String neighbor : adjacencyList.get(vertex)) {
            if (!visited.contains(neighbor)) {
                dfsHelper(neighbor, visited);
            }
        }
    }

    public static void main(String[] args) {
        GraphSearch graph = new GraphSearch();

        // Adding vertices
        graph.addVertex("A");
        graph.addVertex("B");
        graph.addVertex("C");
        graph.addVertex("D");
        graph.addVertex("E");

        // Adding edges
        graph.addEdge("A", "B");
        graph.addEdge("A", "C");
        graph.addEdge("B", "D");
        graph.addEdge("C", "E");
        graph.addEdge("D", "E");

        // Perform BFS and DFS
        graph.bfs("A");
        graph.dfs("A");
    }
}
```

Task - Level 3

Write the program to create shortest path algorithm in java using IntelliJ IDE and Write the program to create the Fibonacci function using recursion in java.

```
import java.util.*;

public class DijkstraAlgorithm {

    private Map<String, List<Edge>> adjacencyList;

    public DijkstraAlgorithm() {
        adjacencyList = new HashMap<>();
    }

    public void addVertex(String vertex) {
        adjacencyList.putIfAbsent(vertex, new ArrayList<>());
```

```
    }

    public void addEdge(String from, String to, int weight) {
        adjacencyList.putIfAbsent(from, new ArrayList<>());
        adjacencyList.putIfAbsent(to, new ArrayList<>());
        adjacencyList.get(from).add(new Edge(to, weight));
        adjacencyList.get(to).add(new Edge(from, weight)); // For undirected graph
    }

    public Map<String, Integer> dijkstra(String start) {
        Map<String, Integer> distances = new HashMap<>();
        PriorityQueue<Vertex> priorityQueue = new
PriorityQueue<>(Comparator.comparingInt(Vertex::getDistance));
        Set<String> visited = new HashSet<>();

        // Initialize distances
        for (String vertex : adjacencyList.keySet()) {
            distances.put(vertex, Integer.MAX_VALUE);
        }
        distances.put(start, 0);
        priorityQueue.add(new Vertex(start, 0));

        while (!priorityQueue.isEmpty()) {
            Vertex currentVertex = priorityQueue.poll();
            if (visited.contains(currentVertex.getName())) continue;
            visited.add(currentVertex.getName());

            for (Edge edge : adjacencyList.get(currentVertex.getName())) {
                String neighbor = edge.getTo();
                int newDist = currentVertex.getDistance() + edge.getWeight();
                if (newDist < distances.get(neighbor)) {
                    distances.put(neighbor, newDist);
                    priorityQueue.add(new Vertex(neighbor, newDist));
                }
            }
        }
        return distances;
    }

    private static class Edge {
        private final String to;
        private final int weight;

        public Edge(String to, int weight) {
            this.to = to;
            this.weight = weight;
        }

        public String getTo() {
            return to;
        }

        public int getWeight() {
            return weight;
        }
    }
```

```
    private static class Vertex {
        private final String name;
        private final int distance;

        public Vertex(String name, int distance) {
            this.name = name;
            this.distance = distance;
        }

        public String getName() {
            return name;
        }

        public int getDistance() {
            return distance;
        }
    }

    public static void main(String[] args) {
        DijkstraAlgorithm graph = new DijkstraAlgorithm();

        graph.addVertex("A");
        graph.addVertex("B");
        graph.addVertex("C");
        graph.addVertex("D");
        graph.addVertex("E");

        graph.addEdge("A", "B", 4);
        graph.addEdge("A", "C", 2);
        graph.addEdge("B", "C", 5);
        graph.addEdge("B", "D", 10);
        graph.addEdge("C", "D", 3);
        graph.addEdge("D", "E", 1);

        Map<String, Integer> distances = graph.dijkstra("A");

        System.out.println("Shortest paths from A:");
        for (Map.Entry<String, Integer> entry : distances.entrySet()) {
            System.out.println("To " + entry.getKey() + " distance is " + entry.getValue());
        }
    }
}
```

Java code for Fibonacci Function

```
public class Fibonacci {

    public static void main(String[] args) {
        int n = 10; // Example: compute the first 10 Fibonacci numbers
        System.out.println("Fibonacci number at position " + n + " is: " + fibonacci(n));
    }

    public static int fibonacci(int n) {
        if (n <= 1) {
            return n;
        }
        return fibonacci(n - 1) + fibonacci(n - 2);
```

```
    }
}
```