EC4070: Data Structures and Algorithms

LAB 06

K.J.M.U.G.S. Eranda Jayasinghe

2021/E/075

SEMESTER 4

EC4070

08.11.2023

FINAL

Q1.

import java.util.Scanner;

class City {

String name;

double latitude;

double longitude;

City left, right;

public City(String name, double latitude, double longitude) {

this.name = name;

this.latitude = latitude;

this.longitude = longitude;

this.left = this.right = null;

}

}

class CityDatabase {

private City root;

public City insert(City node, String name, double latitude, double longitude) {

if (node == null) {

return new City(name, latitude, longitude);

}

if (name.compareTo(node.name) < 0) {

node.left = insert(node.left, name, latitude, longitude);

} else if (name.compareTo(node.name) > 0) {

node.right = insert(node.right, name, latitude, longitude);

}

return node;

}

public void insertCity(String name, double latitude, double longitude) {

root = insert(root, name, latitude, longitude);

}

public City delete(City node, String name) {

if (node == null) {

return node;

}

if (name.compareTo(node.name) < 0) {

node.left = delete(node.left, name);

} else if (name.compareTo(node.name) > 0) {

node.right = delete(node.right, name);

} else {

if (node.left == null) {

return node.right;

} else if (node.right == null) {

return node.left;

}

node.name = minValue(node.right);

node.right = delete(node.right, node.name);

}

return node;

}

public void deleteCity(String name) {

root = delete(root, name);

}

public String minValue(City node) {

String minValue = node.name;

while (node.left != null) {

minValue = node.left.name;

node = node.left;

}

return minValue;

}

public City search(City node, String name) {

if (node == null || node.name.equals(name)) {

return node;

}

if (name.compareTo(node.name) < 0) {

return search(node.left, name);

}

return search(node.right, name);

}

public void searchCity(String name) {

City result = search(root, name);

if (result != null) {

System.out.println("City found: " + result.name + " Latitude: " + result.latitude + " Longitude: " + result.longitude);

} else {

System.out.println("City not found");

}

}

public void printDescendingOrder(City node) {

if (node != null) {

printDescendingOrder(node.right);

System.out.println("City: " + node.name + " Latitude: " + node.latitude + " Longitude: " + node.longitude);

printDescendingOrder(node.left);

}

}

public void printCitiesDescendingOrder() {

printDescendingOrder(root);

}

public void printCitiesWithinDistance(City node, double latitude, double longitude, double distance) {

if (node != null) {

printCitiesWithinDistance(node.left, latitude, longitude, distance);

double dist = calculateDistance(node.latitude, node.longitude, latitude, longitude);

if (dist <= distance) {

System.out.println("City: " + node.name + " Latitude: " + node.latitude + " Longitude: " + node.longitude);

}

printCitiesWithinDistance(node.right, latitude, longitude, distance);

}

}

public void findCitiesWithinDistance(double latitude, double longitude, double distance) {

printCitiesWithinDistance(root, latitude, longitude, distance);

}

public double calculateDistance(double lat1, double lon1, double lat2, double lon2) {

double R = 6371;

double lat1Rad = Math.toRadians(lat1);

double lon1Rad = Math.toRadians(lon1);

double lat2Rad = Math.toRadians(lat2);

double lon2Rad = Math.toRadians(lon2);

double dlon = lon2Rad - lon1Rad;

double dlat = lat2Rad - lat1Rad;

double a = Math.pow(Math.sin(dlat / 2), 2) + Math.cos(lat1Rad) \* Math.cos(lat2Rad) \* Math.pow(Math.sin(dlon / 2), 2);

double c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R \* c;

}

public static void main(String[] args) {

CityDatabase cityDB = new CityDatabase();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\n1. Insert City\n2. Delete City\n3. Search City\n4. Print Cities in Descending Order\n5. Print Cities Within a Distance of a Point\n6. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter City Name: ");

String cityName = scanner.nextLine();

System.out.print("Enter Latitude: ");

double latitude = scanner.nextDouble();

System.out.print("Enter Longitude: ");

double longitude = scanner.nextDouble();

cityDB.insertCity(cityName, latitude, longitude);

break;

case 2:

System.out.print("Enter City Name to delete: ");

String cityToDelete = scanner.nextLine();

cityDB.deleteCity(cityToDelete);

break;

case 3:

System.out.print("Enter City Name to search: ");

String cityToSearch = scanner.nextLine();

cityDB.searchCity(cityToSearch);

break;

case 4:

System.out.println("Cities in Descending Order:");

cityDB.printCitiesDescendingOrder();

break;

case 5:

System.out.print("Enter Latitude of the Point: ");

double pointLatitude = scanner.nextDouble();

System.out.print("Enter Longitude of the Point: ");

double pointLongitude = scanner.nextDouble();

System.out.print("Enter Distance (in kilometers): ");

double distance = scanner.nextDouble();

System.out.println("Cities within the distance of the specified point:");

cityDB.findCitiesWithinDistance(pointLatitude, pointLongitude, distance);

break;

case 6:

scanner.close();

System.exit(0);

default:

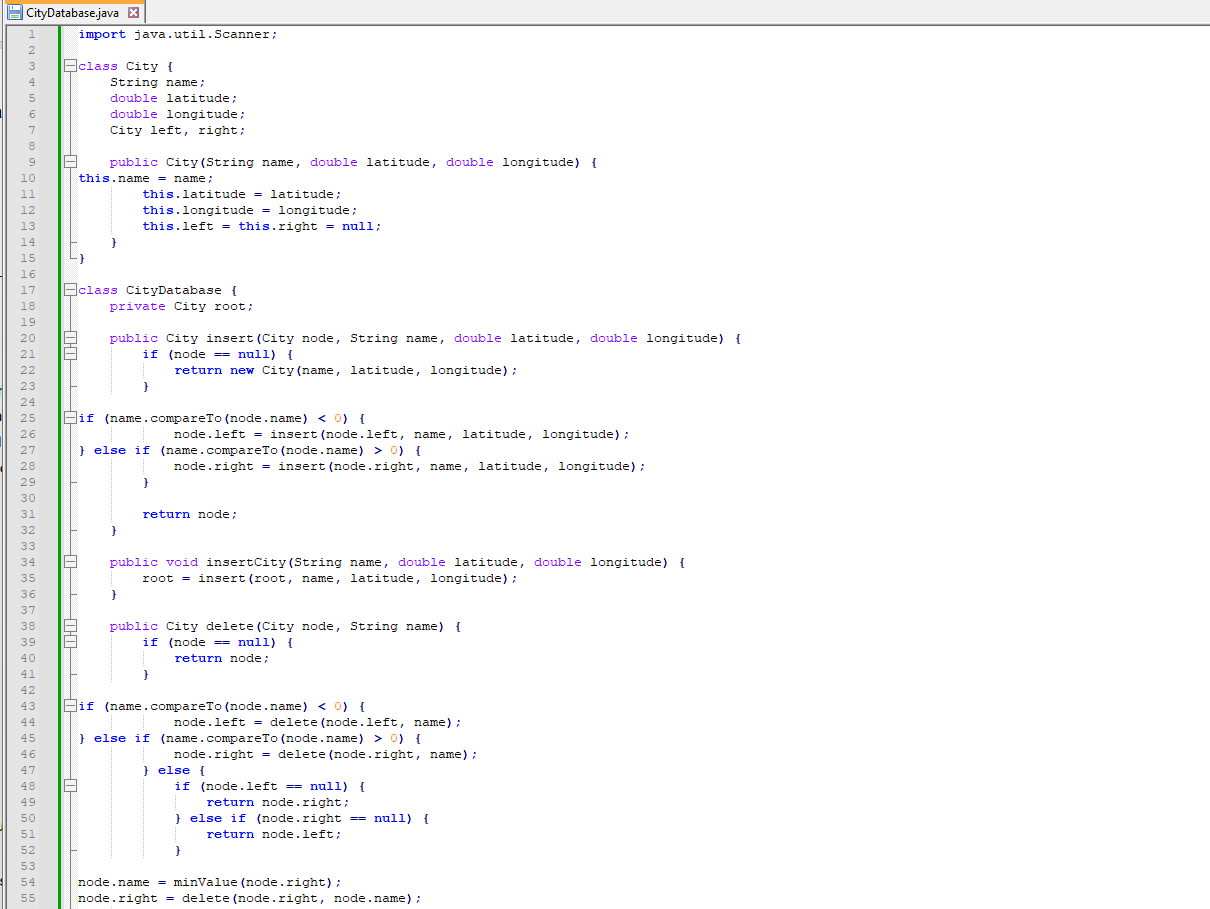
System.out.println("Invalid choice. Please enter a valid option.");

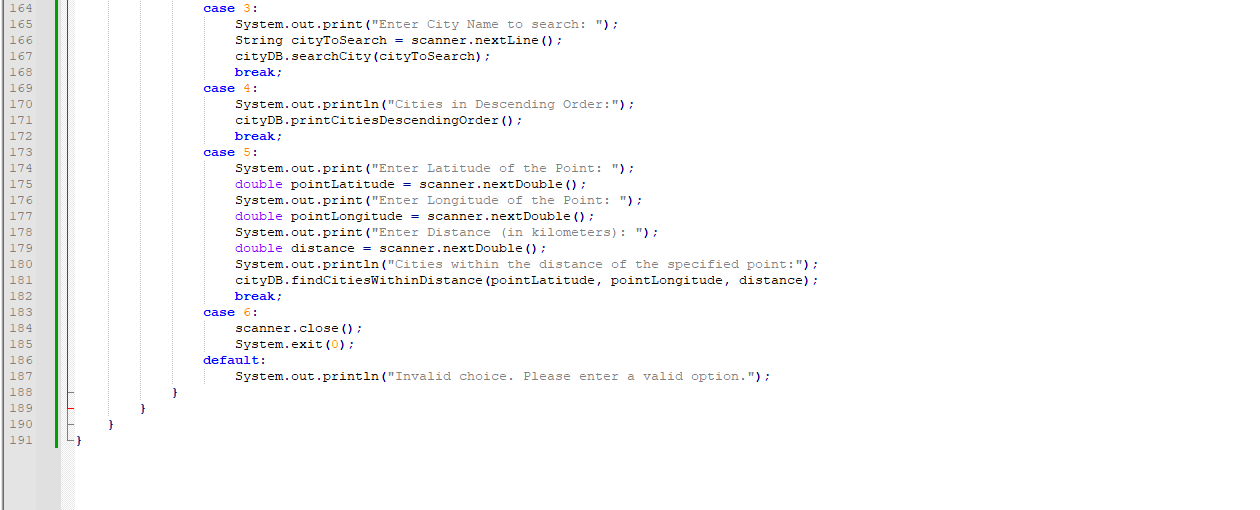
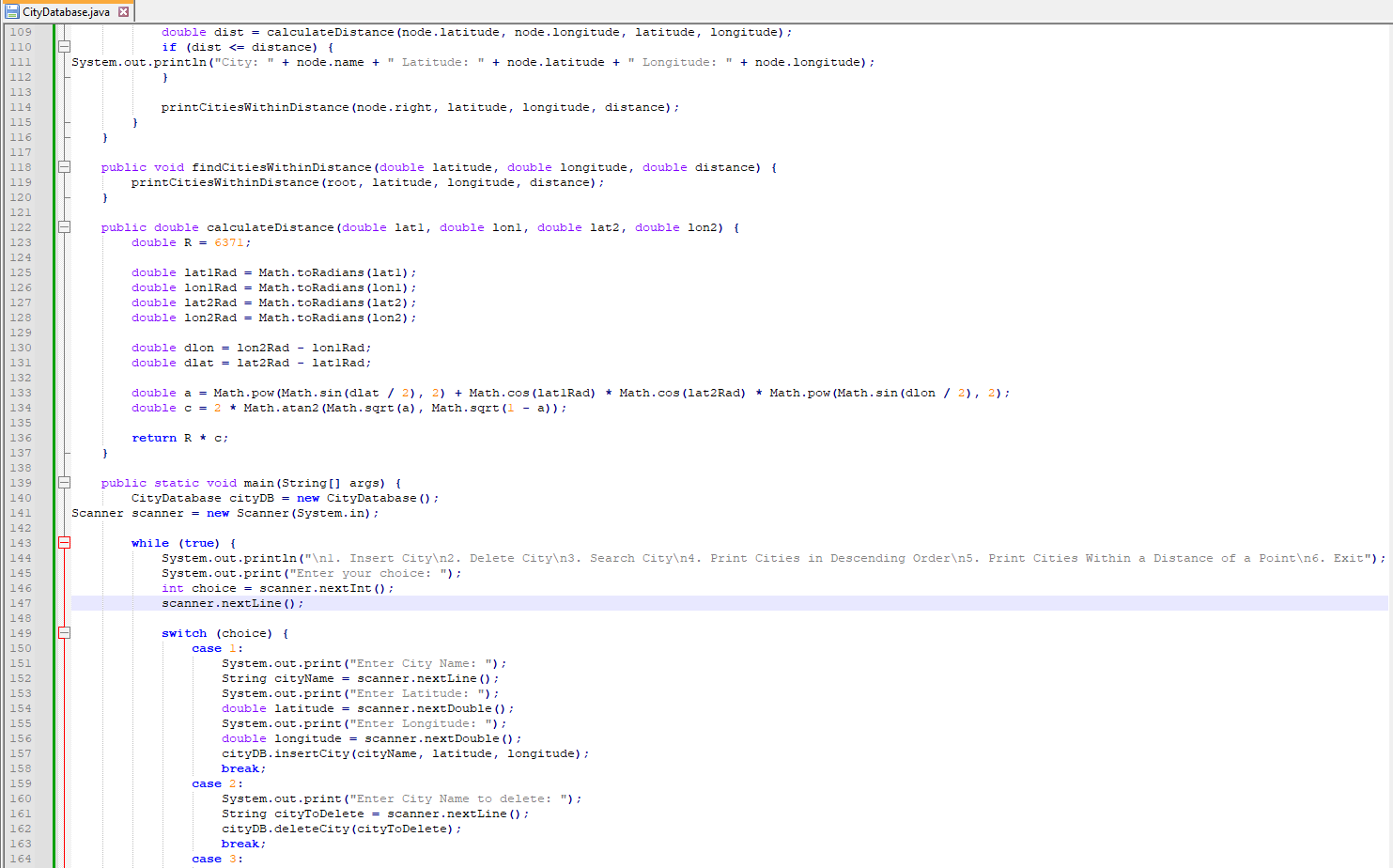
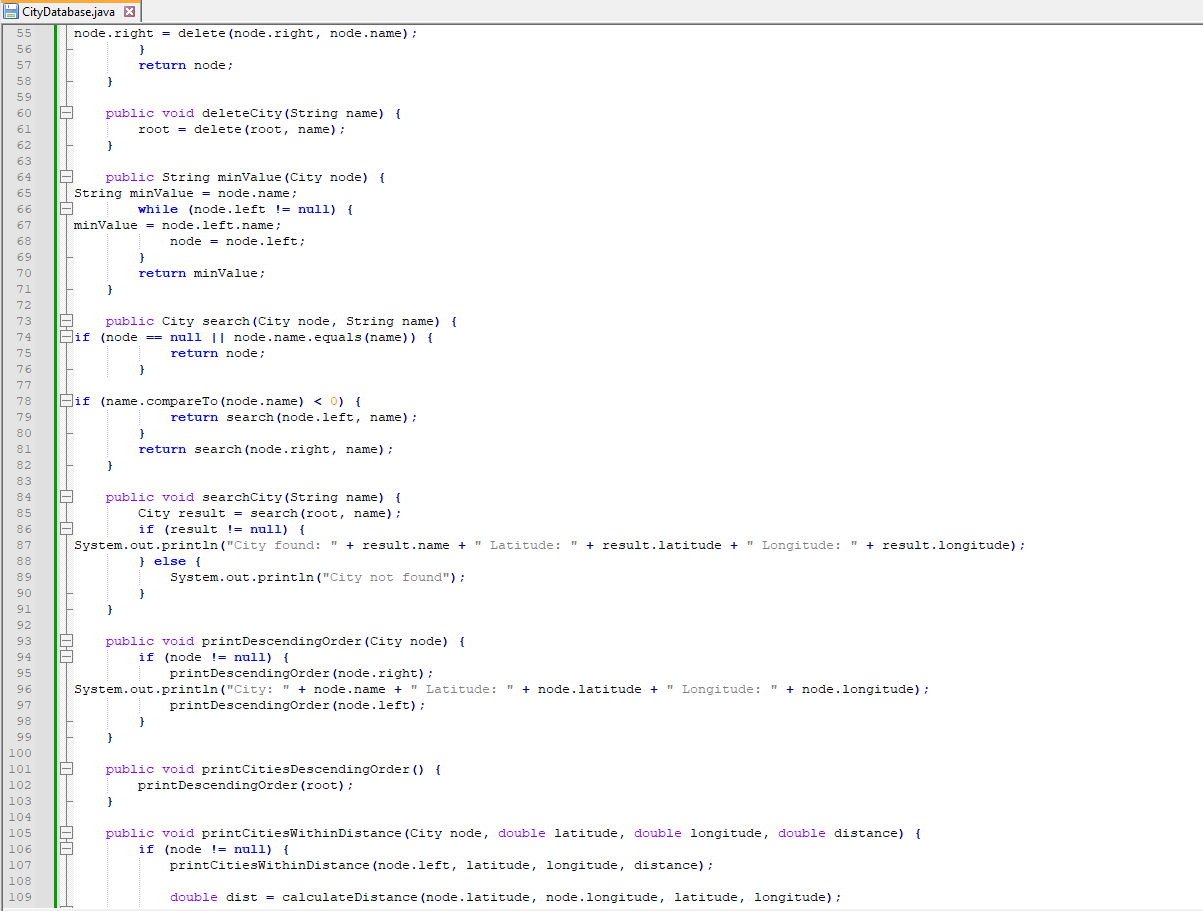
}

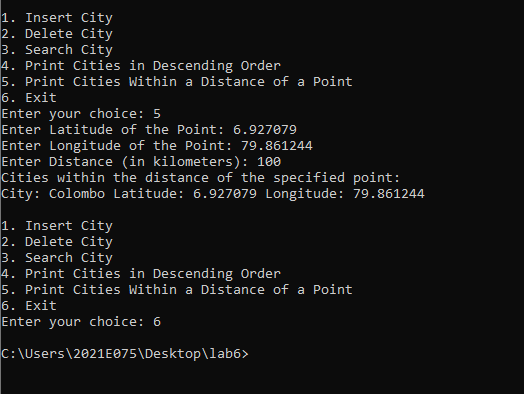
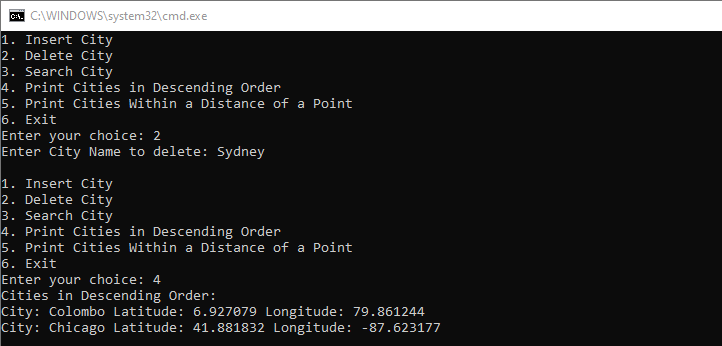
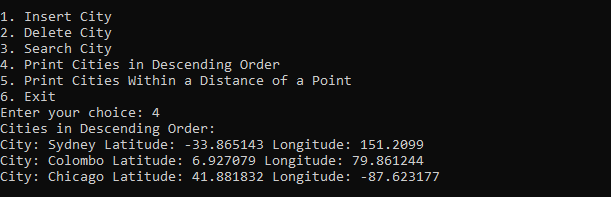
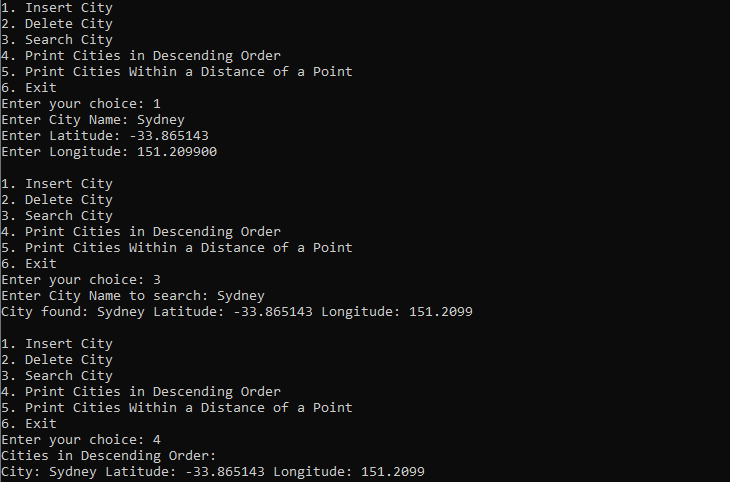
}

}

}







Answer 2

