OOPS Lab

Programming Assignment № 7

Inheritance: Base Classes and Derived Classes

CSE Department, JMI 03/10/2023

Read carefully before you begin:

- Total Marks: 30. Each question carries 10 marks.
- You have 2 hours to complete the assignment. Failure to have your program evaluated before you leave the lab will cause forfeiture of the grade for that lab.
- In order to receive full marks, you must demo the full working code and show the output and given an explanation of your approach where applicable.
- Please **save your code** throughout the semester in a place where you do not lose it. You will be required to submit it at the end.
- Use proper filenaming conventions and commenting. Code that is hard to read or understand will incur a penalty.
- Collaboration must kept to general discussions only. Please do NOT share code or directly share answers with each other. Plagiarism is unacceptable.
- Note: Your Point class must have all of the previous functionality from earlier lab assignments. If it does not, please spend time after today's lab to catch up.

Problem 1 : Derived Classes and the protected access specifier (10 marks)

1. Define a derived class called **Location** that derives publicly from your class **Point**. The derived class should have an attribute called "Address" of type **string**. Do not add any other attributes yet. Add a parameterized constructor that takes three arguments; the spatial coordinates and the address. Implement this derived class.
2. Instantiate an object of type **Location** by default constructor. What is the output? If there are errors, why are they there? Fix the errors (if any). Which constructors get called? Instantiate an object of type **Location** by parameterized constructor using only the coordinates. Which constructor(s) get called? Instantiate using coordinates using coordinates and address. Which constructor(s) get called?
3. Instantiate two **Location** objects and call the **distFrom** function to calculate the distance from the first one to the second one. What is the output? Explain.

sno	name	range	latitude	longitude
1	shahadra mandi	eastern	28.66992	77.29162
2	GTB CROSSING	eastern	28.68991	77.30672
3	kalash nagar pusta	eastern	28.68991	77.25796
4	jagatpuri red light	eastern	28.64789	77.29509
5	ISBT ANAND VIHAR	eastern	28.65005	77.31381
6	Seelampur T-Point	eastern	28.67069	77.26678
7	khajori chowk	eastern	28.7111	77.26034
8	loni gol chakkar	eastern	28.70129	77.29146
9	Kondli Bridge	eastern	28.61812	77.32086
10	nirman vihar	eastern	28.63583	77.28713

Table 1: Location addresses, range and coordinates.

Problem 2: Derived Class Constructors (10 marks)

1. Define and implement all constructors and a default destructor for the class **Location** over-riding the constructors in the class **Point**. Pass the appropriate variables down to the constructors in **Point** by using the single colon ":" operator

2. Override the function **distFrom** in the **Location** class and let it compute the geodesic distance between two locations instead of the Euclidean distance.

   *Hint: The simplest way to calculate geodesic distance is to find the angle between the two points, and multiply this by the circumference of the earth. The formula is: angle = arccos(point1 * point2) distance = angle * pi * radius. Radius of the Earth ≈ 6,371 km*

3. Let's say, a delivery truck has to start from location 1 on the list given in Table 1 and go to location number 10 while stopping at each location in the given order. What is the total distance it must travel in kilometers?

Problem 3: Abstract classes and virtual functions (10 marks)

1. Declare an abstract class called **Element** with a pure virtual function called **print**.

2. Modify the **Point** and **Vector** classes in your code to inherit from the class **Element** publicly. Provide an implementation for the **print** function for each one of them.

3. Speculate, if you can you call the **print** function from an object of type **Location**. Why or why not? Try it and see. What is the result?

1 [OPTIONAL] Problem 4: Shortest Path

Let's say the adjacency matrix for the DAG connecting the locations in Table 1 is given by the following:

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	1	0	0	0
3	0	0	0	0	0	1	0	0	0	1
4	0	0	0	0	0	0	0	1	0	0
5	1	0	0	0	0	1	0	0	0	0
6	0	0	0	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1
8	1	1	0	0	0	0	0	0		0
9	0	0	0	0	0	1	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0

Check if the graph represented by the matrix has cycles. Implement a function called **shortestPath** that uses the Bellman-Ford Algorithm to compute the shortest path from one location to every other location in a directled acyclic graph. Instruct the driver who starts out at location 1 how to proceed to all other locations using a greedy approach. Is he able to complete his journey?