ICS 202 Lab Project (Sections 51, 52, 53, and 55) – Term 202

Guidelines:

1. This is a strictly individual project. No collaboration is allowed.

2. Due date for the project is 26th April 2021 at 23:59.

3. You also need to include a report which should have the following:

title page with (a) project name, (b) your name and ID.

listing of all source code of the project in readable format (Consolas font size 10 is preferred)

output screen shots of the project.

4. Submit the report and the java files in a zip file named in the format:

LabSectionNumber\_LabProject\_KFUPMID#\_FamilyName

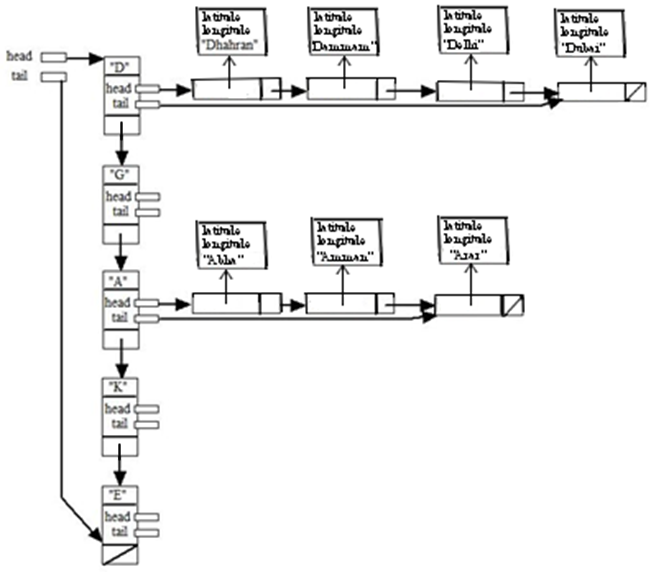
Example:

**53\_LabProject\_201700000\_AlHarthi**

Note: Please do not attach any characters to the section number

Implement the following data structure that stores city objects according to city sub-lists:

* Each sub-list is an instance of the given SLL<T> class
* The main list (MSLL<T>) is a modified SLL whose nodes have the following fields:
  + **info** field of generic type T
  + **list** field of type SLL<T>
  + **next** of type MSLL<T>



A city object has the fields:

* cityName of type String
* decimalLatitude of type double
* decimalLongitude of type double

It also has appropriate methods (that you are required to provide)

**Note**: Each citySublist contains the city objects whose names start with the string in the info field of the corresponding MSLL node field.

1. Implement the **City** class
2. Implement the **MSLL** node class
3. Implement the **SLL** node class
4. Implement each of the following methods of **SLL** or **MSLL** (Note you are required to determine which method is appropriate for the two types of lists) :

* **public void addCityToSublistAtRear (SLL CLASS)**

Adds a city object at the end of a sublist. Throws exception if city exists in sublist.

* **public void addToCitySublistList(T e1, int position) (SLL CLASS)**

Adds a city object at a given position of a city sublist. Throws exception if city exists in sublist

or if position is not valid

* **public boolean cityIsInSubList(T e1) (SLL CLASS)**

Returns **true** if city object is in sublist, otherwise it returns **false**.

* **public void deleteCityFromSublist(T e1) (SLL CLASS)**

Deletes city object if present in sublist; otherwise it throws an exception it is not present.

* **public void deleteCitySublist(String str) (MSLL CLASS)**

Deletes the MSSL node with **str** as key. Throws an exception if such a node does not exist.

* **public void makeCitySublistEmpty(String str) (SLL CLASS)**

Makes the city sublist of the MSSL node with **str** as key empty. Throws an exception if such a node

does not exist.

* **public void displayCitySublist(T e1) (SLL CLASS)**

Displays all city objects in the city sublist corresponding to MSLL node with key e1.

Throws an exception if such a node does not exist.

* **public void addToMSLLHead(T el) (MSLL CLASS)**

Adds a new node with key **e1** at the beginning of the MSLL list if a node with this key does

not exist; otherwise, it throws an exception.

* **public void addToMSLLTail(T el) (MSLL CLASS)**

Adds a new node with key **e1** at the end of the MSLL list if a node with this key does not

exist; otherwise, it throws an exception.

* **public void deleteFromMSLL(T el) (MSLL CLASS)**

Delete the MSSL node with key e1 if it exists. Throws an exception if MSLL is empty or if

Node with key **e1** does not exist.

* **public boolean isInMSLList(T el) (MSLL CLASS)**

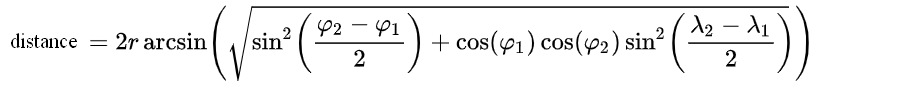
Returns **true** if MSLL has a node with key **e1**; otherwise, it returns **false**.

* **public double getDistance(String city1, String city2)**

Calculates the straight-line distance between city1 and city2 using **Haversine** formula.

Throws an exception if any of the two cities does not exist.

Harvesine formula:



Where:

* ***distance***is the straight-line distance in kilometers between the two points along a great circle of the earth sphere,
* *r* is the radius of the earth sphere = 6372.8 kilometers.
* *φ*1, *φ*2 are the latitude of point 1 and latitude of point 2 (in **radians**),
* *λ*1, *λ*2 are the longitude of point 1 and longitude of point 2 (in **radians**).

**Note**: Write a test class to test the methods of SLL and MSLL. The test program must be menu driven with the following menu options:

1. AddCityToSublistAtRear
2. AddCityToSublistAtPosition
3. SearchForCity
4. DeleteCityFromCitySublist
5. DeleteCitySublist
6. MakeCitySublistEmpty
7. DisplayCitySublist
8. addToMSSLAtHead
9. AddToMSSLAtTail
10. DeleteFromMSSL
11. SearchMSLList
12. getDistance
13. Exit

Your program must:

* Throw an exception for any wrong menu choice.
* Loop and display the menu again as long as the choice is not 14.
* Provide appropriate behavior for each option.

Sample program run, when the **getDistance** option is selected and both **Dhahran** and **Mecca** are present in the data structure:

Enter the name of city1: **Dhahran**

Enter the name of city2: **Mecca**

The straight-line distance between Dhahran and Mecca is 1181.76 kilometers

Sample program run, when the **AddCityToSublistAtPosition** option is selected and the entered city object is not in the corresponding sublist and position is valid:

Enter the city name: **Amman**

Enter the city latitude in decimal degrees: **31.9539**

Enter the city longitude in decimal degrees: **35.9106**

Enter the insertion position: **1**

Some cities and their decimal latitudes and longitudes:

|  |  |  |
| --- | --- | --- |
| City | latitude | longitude |
| Abha | 18.2167 | 42.5 |
| Amman | 31.9539 | 35.9106 |
| Arar | 30.9833 | 41.0167 |
| Amsterdam | 52.3676 | 4.9041 |
| Arusha | -3.3869 | 36.6830 |
| Dhahran | 26.2361 | 50.0393 |
| Dammam | 26.4333 | 50.1000 |
| Delhi | 28.7041 | 77.1025 |
| Dubai | 25.2048 | 55.2708 |
| Edinburgh | 55.9533 | -3.1883 |
| Entebbe | 0.0512 | 32.4637 |
| Gaborone | -24.6282 | 25.9231 |
| Genoa | 44.4056 | 8.9463 |
| Glasgow | 55.8642 | -4.2518 |
| Riyadh | 24.6500 | 46.7100 |
| Rome | 41.9028 | 12.4964 |
| Rotterdam | 51.9244 | 4.4777 |
| ReoDeJaneiro | -22.9068 | -43.1729 |
| Jeddah | 21.5428 | 39.1728 |
| Jizan | 16.8894 | 42.5706 |
| Jakarta | -6.2088 | 106.8456 |
| Jubail | 26.9598 | 49.5687 |
| Jalalabad | 34.4284 | 70.4578 |
| Johannesburg | -26.2041 | 28.0473 |
| Khobar | 26.2172 | 50.1971 |
| Kabul | 33.9391 | 67.7100 |
| Karachi | 24.8607 | 67.0011 |
| Khartoum | 15.5007 | 32.5599 |
| Mecca | 21.4225 | 39.8262 |
| Medina | 24.4667 | 39.6000 |
| Madrid | 40.4168 | -3.7038 |
| Manama | 26.2235 | 50.5876 |