Curtin College-Singapore
DSA1002 Data Structure and Algorithm
Trimester 3, 2022

Assignment 1

Weight: 30% of the unit

Assignment Location: The assignment 1 is uploaded under **Assessments section (Assessment 2 – Assignment 1)** on unit Moodle page**.**

Answer Format. When you write an answer, clearly indicate the relevant question number/letter. Include your name and student ID at the start. Also add appropriate comments to code files to indicate author name and student ID. Detailed submission guidelines can be found below in section 3.

Timeframe. You have 14 days (336 hours) to complete and submit your answers, from **08:00am on 19th Dec 2022** until **09:00am on 09th Jan 2023** (UTC+8). You may schedule your work within this period. **However, late submissions are not allowed** (also check late submission policy in unit outlines).

Submission. Submit your answer document(s) to the "**Assessment 2 – Assignment 1**" area on Moodle under assessments section. You must verify that your submission was successful. Correctly submitting is entirely your responsibility.

Reference Material. This is an OPEN BOOK and OPEN COMPUTER assignment. You may refer to any written material, including your notes, course materials, books, websites, Unit Moodle page recordings etc. However:

- You must complete this assignment entirely on your own.
- You should answer all questions in your own words and code.
- You can use pseudo code and algorithms provide in the unit slides (Moodle page) for your implementation.
- During the assignment, you may not communicate with any other students/anyone helping.
- Your answer document will be checked by text matching software for signs of cheating,collusion and/or plagiarism.
- The assignment questions have been designed such that any two students, working independently,should not produce the same answers.
- The coding part of this assignment can be submitted in either python/java.

****** READ COMPLETE DOCUMENT BEFORE STARTING******

1. Overall Assignment Description

In this assessment you will apply a detailed knowledge of data structure and algorithms to real-life application to have a better understanding of the details covered in the unit. Detailed description on each question and steps required to be performed can be found in the question description. You can use pre-built ADTs from your tutorial and workshops (should be appropriately cited). Do not use build-in programming language (Python/Java) ADTs, e.g., using built-in Python stack.

Question 1: (Mini Project)

This assignment is an extension of the code you implemented for graph implementation in the tutorial session. You will be implementing map of Curtin University. You are supposed to develop the algorithm to find the path from staring point to end point on the map. The code you will implement should have following properties/implementation details.

*All code should be implemented as dynamic i.e., based on linked lists.

Maze implementation (ADT code)

1. The code should be implemented as an ADT called CurtinMap, this may require you to implement multiple classes (e.g., vertex, edge, graph, as for class graph implementation).

Hint: The classes we implemented may remain same i.e., vertex (storing label of location, value (if required), edge (label, direction and weight), and graph (defining map and related operation, adding nodes, finding paths. Best way is to design your UML diagrams, this also help you in report of the assessment.

2. The required methods should be implemented in the relevant classes. The following should be supported by the implemented map added.
    a. AddLocation, add vertices provided in the file "Vertices.csv".
    b. AddPath, add edges provided in the file "Edges.csv".
    c. PrintAllLoc (printing list of locations in the map)
    d. PrintAllPaths (printing a list of all paths between any two locations with distance i.e., weight of the path)
    e. FindPath (edit DFS to run until you reach to the destination location from start location). This function should also print total distance of the path (i.e., accumulative weight of all visited paths/edges).
    f. MinPath (execute DSF and BSF from staring to endpoint, and print the path with smaller distance only, use the concept described in (e))

Hint: e.g., AddVerted: this function can be used to add vertices to the graph (i.e., locations in the map), AddEdge: this function can be used to add edges/path between vertices/nodes. Other important methods could be printing all paths (print all edges), print all locations (print all vertices), etc.

*All code should be implemented using your own designed ADT only. Cannot use any python built-in support for lists/stacks/queue, etc. You can find reference from lecture and tutorials how we implemented there. You can use your built codes for tutorials/practicals/previous submissions (should be referenced).

Menu (test harness)

The co should be implemented as console applications/interactive code. Following options should be considered foe the menu.

1. Add new location(s) (read the file provided to add locations to the map)
2. Add path(s) (read file to add paths between the inserted locations)
3. Print all locations
4. Print all paths (print path name/label distance connections i.e., the path is between which two locations)
5. PathBetween (print path between two given locations with total distance/weight)
6. PathMin (Smallest path between locations)

The menu should be interactive and may ask user when to exit the menu.

2 Project Report

A project report of minimum 8-10 pages should be submitted (pdf format) including following details:

Usage information:

- Introduction: describing basic introduction of your program (software).
- Dependencies: any libraries required to use the program (software).
- Terminologies and abbreviations used in the code
- Future directions: suggested future improvements

Class UML Diagrams:

- Readme file, describing related information[1].
- Complete UML class diagrams of the classes used for implementation.
- A complete association of classes/objects (i.e., class relationship).
- Complexity analysis of all operations performed by the software (e.g. AddPath, AddLocation, etc.).
- Traceability matrix of feature implementation and testing of your code[2].

Comments on Code: it is suggested to add detailed comments in your code.

References: (if any) all materials should be referenced Chicago referencing style.
[1]Example good Readme files can be found here.
[2]Traceabiliy matrix help.

3 Submission

Submit electronically through Moodle unit page under assessments section ("**Assessment 2 – Assignment 1**").
You should submit a single file, which should be zipped (.zip) or tarred (.tar.gz). Check that you can decompress it on the lab/personal computers. Your work will be tested on lab/computer other than your PC so try to check your code on other PCs too. The file must be named DSA_Assignment_1_<student id>, use underscores instead of the spaces in the file name.
The file should contain following deliverables:

- Your code. This means all.java/.py files needed to run your program. Do include code provided to you as part of the assignment if that is required to run your program. Do not include .class files or anything else that is not required to recompile .java files.

- README file including short descriptions of all files and dependencies, and information on how to run the program (see section 2).
- Your program (software) test harnesses. One of the easiest ways for us to be sure that your code works is to make sure that you've tested it properly.
- Documentation and Report for your code (Project Report)

Please verify that your submission is correct and not corrupted. You may make multiple submissions, only your last one will be marked. **However, late submissions are strictly not allowed** (also check late submission policy in unit outlines).

4 Marking Criteria

The assignment will be marked based on the following breakdown of the submission:

Code Demo: (40 Marks) Code should be demonstrated during the tutorial to achieve this requirement. The code should be appropriately written, as ADTs with comments. The code developed will be tested against different tests (as per requirements given in Questions 1).

Project Report: (30 Marks) A minimum 8-10-page report based on information describe in section 2.

Code Testing: (30 Marks) Code should be implementable, and testable with the test harness.

5 Academic Integrity

Please see the Coding and Academic Integrity Guidelines on unit Moodle page.

In summary, this is an assessable task. If you use someone else's work or assistance to help complete part of the assignment, where it's intended that you complete it yourself, you will have compromised the assessment. You will not receive marks for any parts of your submission that are not your own original work. Further, if you do not reference any external sources that you use, you are committing plagiarism and/or collusion, and penalties for academic misconduct may apply.

Curtin college also provides general advice on academic integrity at https://www.curtincollege.edu.au/content/dam/navitas/upa/curtin/pdfs/academic-integrity-policy.pdf

The unit coordinator may require you to provide an oral justification of, or to answer questions about, any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an academic misconduct inquiry.