Data Structures and Algorithms

Assignment v1.1

Semester 1, 2021

Department of Computing

Curtin University

1 Introduction

In practicals you have implemented and learned about a number of algorithms and ADTs and will be implementing more of these in the remaining practicals. In this assignment, you will be making use of this knowledge to implement a system to explore and compare a variety of ADT implementations. Feel free to re-use the generic ADTs from your practicals. However, remember to self-cite; if you submit work that you have already submitted for a previous assessment (in this unit or any other) you have to specifically state this. Do not use the Java/Python implementations of ADTs – if in doubt, ask.

2 The Problem

This assignment requires the development of a program to model a series of shops or warehouses to be visited by a drone to collect items. You will have a "purchase order" of items to collect, that could be at any of the locations. Your program should read in the inventories (stock on hand) at each location, and store this using appropriate ADTs. The distances between locations will be given, and your decision on which locations to visit to complete the order can be optimised for the shortest distance. A list of unfilled items should be given if you cannot complete the order.

Your program should be called the **droneCollector** and have three starting options:

- No command line arguments : provides usage information
- "-i" : interactive testing environment
- "-r" : report mode, usage:
  `droneCollector -r <location_file> <prod_file> <inventory_file> <order_file>`

When the program starts in interactive mode, it should provide the following menu options:

(0) Load data a. Location data b. Inventory data c. Product data d. Order data e. Serialised data	*- should be an initial menu to ensure that required data is loaded in correct order*
(1) Location overview	*Listing information about all locations*
(2) Inventory overview	*Listing of all location inventories*
(3) Product search	*Give details for a particular product in the*

	catalogue
(4) Find and display inventory for a location (store)	*e.g. "Jaycar"*
(5) Find and display distance between two locations	*e.g. "Jaycar Apple'*
(6) Find and display route for collecting order	*- Given pre-loaded order, print locations in order and the items to collect from each*
(7) Save data (serialised)	*All data can be saved to avoid re-reading all input files and reconstructing data structures*
(8) Exit	

You can structure the menu/UI differently, just make sure at least the specified options are included.

When running in report mode, you will give the input files and any parameters on the command line, then output the route / locations / items (menu item 6).

You will then investigate the performance of the code you have developed. This investigation will be written up as the Project Report.

Remember: think before you code!

3 Submission

Submit electronically via Blackboard.

You should submit a single file, which should be zipped (.zip) or tarred (.tar.gz). Check that you can decompress it on the lab computers. These are also the computers on which your work will be tested, so make sure that your work runs there. The file must be named DSA_Assignment_<id> where the <id> is replaced by your student id. There should be no spaces in the file name.

The file must contain the following:

- **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.
- **README** file including short descriptions of all files and dependencies, and information on how to run the program.
- Your **unit 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. Make it easy for us to test your work - the test harness for class X (or module X) should be called UnitTestX.
- **Documentation and Report** for your code, as described in Section 3.1.
- A signed and dated **cover sheet**. These are available from Blackboard with the assignment specification. You can sign a hard copy to scan or fill in a soft copy and digitally sign it.
- **Java Students:**
    - Do not include .class files or anything else that we do not need. We will recompile .java files to ensure that what we're testing is what we're reading. We will use javac *.java to compile your files and run the unit tests by their expected names.

Make sure that your file contains what is required. Anything not included in your submission will not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct.

3.1 Project Report

Please submit the Project Report in PDF format. Your Report will be minimum 8-10 pages (excluding UML and Javadocs) and should include the following:

Information for Use	• **Introduction:** Briefly discuss the functionality of your program • **Installation:** requirements, dependencies & description of files/directories • **Terminology/Abbreviations:** • **Walkthrough:** Demonstrate all functionality, indicating anything that isn't implemented or working properly • **Future Work:** Missing items or suggested enhancements
Traceability Matrix	Map the requirements in the specification to the implementation and tests – indicating whether they have been done and where to find more information (Class Descriptions, Justification and/or Information for Use).
Class Diagram	A UML class diagram of your code
Class Descriptions	A description of any **classes** you have, you need to let us know not only what the purpose of that class is but why you chose to create it. As part of this, also identify and justify any places where it was possibly useful to create a new class but you chose not to, especially when it comes to inheritance.
Justification	Explain any questions that the marker may have. This is supplemented by the comments in your code – however marks are allocated for the information being in the Project Report. In particular, when you choose an ADT, underlying data structure or an algorithm, you need to justify why you chose that one and not one of the alternatives. You should include some prediction of the expected "performance" of your code – this includes time complexity and/or memory usage. Include the commands, input files, outputs – anything needed to reproduce your results.
References	Chicago referencing style

3.2 Marking

Marks will be awarded to your submission as follows:

Code demonstration [30 marks]	We'll have a number of tests to run, and you get marks proportional to how many tests your code passes. If your code passes all of the tests, then you will get all of these 30 marks.
Project Report [40 marks]	PDF or Word document - see Section 3.1
Implementation [30 marks]	We will use unit testing as well as looking at code quality; your test harnesses will have a big impact on these marks. Students are encouraged to use testing frameworks for their harnesses – don't

	stress about it though… DIY test harnesses are fine.
Bonus Marks [5 marks each]	There will be bonus marks available exceptional/additional features. Examples of enhancements could include: visualisation and plotting, optimising the route etc.

- Marks will be deducted for not following specifications outlined in this document, which includes incorrect submission format and content and using built-in Java/Python ADTs.
- If the cover sheet isn't provided with your submission, your submission will not be marked and you will be awarded zero (0) marks. If you forget to submit the cover sheet you will be allowed to submit it separately to the unit coordinator but will lose 5 marks.

3.3 Requirements for passing the unit

Students must submit an assignment worthy of scoring 15% to pass the unit.

This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au.

In the case of doubt, you may be asked to explain your code and the reason for choices that you have made as part of coding to the unit coordinator. A failure to adequately display knowledge required to have produced the code will most likely result in being formally accused of cheating.

Finally, be sure to secure your code. If someone else gets access to your code for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

3.4 Late Submission

Late submissions will incur a 10% deduction per day.

3.5 Clarifications and Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced via Blackboard. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks.