Data Structures and Algorithms

Assignment

Semester 2, 2018 v1.0

Department of Computing
Curtin University

1 Preamble

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 that is somewhat more open ended. In this system, you will be deciding (for the most part) how to structure the program, how to use the abstract data types and your program will need to decide on the flow of objects in and out of them. 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 implementations of ADTs – if in doubt, ask.

2 The Problem

It is election time 2016. Australia is voting for a new government and Prime Minister to take us through the next 3 years... The campaign management team need to get overall information about their position in the lead-up to the election (we will use the 2016 election results as the "poll data"). They also need to know which divisions should be visited to get maximum impact, and provide an itinerary based on the distances/time between the locations. This scenario should allow you to reuse parts of many of your practicals, which you will self-cite.

The source data can be downloaded from the Australian Electoral Commission website at: https://results.aec.gov.au/20499/Website/HouseDownloadsMenu-20499-Csv.htm. You will need (at least) the files for the National List of Candidates and First Preferences by Candidate by Polling Place. Polling place data should be aggregated into Divisions. We will provide data for the distances between divisions and announce it on Blackboard.

When the program starts up, it should load the input data and then show the following main menu:

```
(1)     – List Nominees
(2)     – Nominee Search
(3)     – List by Margin
(4)     – Itinerary by Margin
(0)     – Quit
Choice:>
```

The user enters in their choice and the system executes the desired option (possibly involving further user input). Reports will be displayed to screen with an option to save to file. After the option has completed processing, the main menu should be re-displayed. In other words, make each option its own method, and call this from a mainMenu() (or similar) method that executes a loop until the Quit option is made.

Reports:

- List Nominees :
    - Give list of nominees, giving options to filter by (any/all of) State/Party/Division and to order by (any/all of) Surname/State/Party/Division.
- Nominee Search :
    - Give full details of nominees, searching by a substring, starting with the first character of the surname, giving options to filter by (any/all of) State/Party.
- List by Margin :
    - Select a party and list all marginal seats within a threshold. Allow default values of +/-6%, as well as entering custom margins.
- Itinerary by Margin :
    - Based on a List by Margin, select locations to visit and create an itinerary to visit all the locations. Assume they need three hours per location to meet/greet and promote. Additional marks awarded for minimising on overall travel time.

3 Submission

Submit electronically via Blackboard. Make sure to submit early. You can submit multiple times – we will only mark the last attempt. Take care not to submit your last version late though. Read the submission instructions very carefully.

You should submit a single file, which should be zipped (.zip) or tarred (.tar.gz). Check that you can decompress it on lab machines. These are also the computers that your work will be tested on, 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; use underscores as shown.

The file must contain the following:

- Your code. This means all files needed to run your program. Do not include any .class files. Do include input files used 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 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. Our test harnesses may not work for some of your classes and you may have classes that we're not specifically asking for, so make it easy for us to test your work. A test harness for class/module X should be called UnitTestX/ModuleTestX.
- Documentation for your code, as described in Section 3.1.

- A signed and dated cover sheet. These are available from the [[Computing Colloquium]] Blackboard unit or from the Computing reception (building 314, 3rd floor). You can sign a hard copy and scan it in or you can fill in a soft copy and digitally sign it.
- Java students:
    - You may include a Makefile, but this is not required. We will use javac *.java to compile your files and run the unit tests by their expected names.
    - Do not include .class files or anything else that we do not need. We will compile your .java files to ensure that what we're testing is what we're reading.

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 Documentation

You need to submit documentation in docx or pdf format. Your documentation should include the following:

- An overview of your code, explaining any questions that the marker may have. This is supplemented by the comments in your code. In general, if the marker is looking at your code and isn't sure why you have done something in that particular way, they will check your documentation to see if they can find an explanation. Using an automated documentation system like Javadocs may be very helpful. It is not required, though.
- A UML diagram of your code
- 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 of all major decisions made. 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. These decisions are going to be of extreme importance in this assignment.

3.2 Marking

Marks will be awarded to your submission as follows:

- **[40 marks] Decisions made.** This is mainly things like choosing the right class breakdown, ADT, underlying data structure and algorithm. Note that you will only get these marks if you adequately justify your decisions and properly implement them. So, for example, if you choose an algorithm and adequately justify it but aren't actually able to implement it you'll only get part of the marks. Any justification should be presented in the documentation, which means that your documentation is worth 40% of your marks for this assignment. Start documenting early.
- **[30 marks] Code testing.** 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.
- **[30 marks] Implementation**. We will use unit testing as well as looking at code quality; your test harnesses will have a big impact on these marks.
- Marks will be deducted for not following specifications outlined in this document, which includes incorrect submission format and content and using built-in Java 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 (by e-mail or in person) but will lose 5 marks.

The aims of this marking breakdown is as follows:

- To reward you for good design decisions, but not unless you're able to implement them. This is the core aim of the unit.
- To let you score some marks if you get the program working but make with poor decisions. If all of your decisions are based only on ease of implementation you can still score the 30 marks from the implementation category and will score most of the marks from testing, meaning you can pass if all of your code works perfectly and is of sufficient quality. It's taking a risk, though.
- To promote good testing. Industry is repeatedly telling us they are really looking for students who can properly test their code.
- To teach you to follow specifications carefully.
- To reward students who have been serious about doing their practical work.

3.3 Requirements for passing the unit

Please note: As specified in the unit outline, it is necessary to have attempted the assignment in order to pass the unit. As a guide, you should score at least 15% to be considered to have attempted this assignment. We have given you the exact mark breakdown in Section 3.2. Note that the marks indicated in this section represent maximums, achieved only if you completely satisfy the requirements of the relevant section.

Plagiarism is a serious offence. 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

I know for the vast majority of you, I don't need to tell you what is in this section. But every semester, someone always tries to see what they can get away with here. For the benefit of fairness to the vast majority of you who do the right thing, we need to make the rules surrounding late submission clear

As specified in the unit outline, you must submit the assignment on the due date. Acceptance of late submissions is not automatic and will require supporting documentation proving that the late submission was due to unexpected factors outside your control. See the unit outline for details as to the procedure for requesting that an assessment be accepted after the due date.

Note that external pre-scheduled commitments including, but not limited to, work, travel, scheduled medical, sporting, family or community engagements are not considered unexpected factors outside your control. If you know you have, or are likely to have, such engagements and that they may affect your ability to complete the assignment, you will be expected to have planned your work accordingly. This may mean that you need to start and/or complete your assignment early to make sure that you are able to hand it in on time.

Also note that IT related issues are almost never a valid excuse. These include computer crashes, hard disk corruption, viruses, losing computers/storage media, networks going down (even if it is Curtin's network - outages of the entire Curtin network of more than 24 hours may be considered depending on circumstances) or the like. As IT professionals in training, you are expected to have suitable backups and alternative ways of getting your assignment completed in the event that any IT problems are encountered. You are also expected to submit your assignment ahead of time to allow for unforeseen issues.

In the event that you submit your assignment late and are deemed to have a valid excuse, you will be penalised 10% (that is, 10% out of 100%, not out of what you would have received) per calendar day that you are late, up to a maximum of seven (7) calendar days. Any work submitted after this time will not be marked and you will automatically fail the unit. Note that if you are granted an extension you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

Note that the requirements for passing this unit are applied after penalties. An assignment normally scoring 20% that is submitted one day late, even with a valid excuse, will score only 10% and thus not satisfy the requirements for passing this unit.

3.5 Clarifications and Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced in the lecture and on the unit's Blackboard page (not necessarily at the same time and not necessarily in that order). These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these, either by attending the lectures, watching the iLecture and/or monitoring the Blackboard page.