COMP6008 DATA STRUCTURES AND ALGORITHMS

Summary

Title	Assessment 1 – Programming tasks: applications of data structures and algorithms
Type	Portfolio
Due Date	Friday 13 September 11:59 pm AEST (end of Week 3)
Length	NA
Weighting	30%
Academic Integrity (See below for limits of use where GenAI is permitted)	You **may use** GenAI tools to get some ideas or insight about particular problems in code. However, you **MUST NOT** generate a complete solution code or write a report. Please refer to the **Academic Integrity section** below to see what you can and cannot do with the GenAI tools.
Submission	A word document submitted to Turnitin using the provided template.
Unit Learning Outcomes	This assessment maps to the following ULOs: • ULO1: design and implement common data structures and algorithms • ULO4: apply data structures and algorithms to solve a real-world problem

Rationale

Data structures and algorithms are fundamental skills for computer scientists. The ability to apply common data structures and design efficient algorithms to solve real-world problems often indicates a difference between computing and IT graduates. This assessment aims to assess students' knowledge of data structures and algorithms and skills in applying them to solve problems. It addresses unit learning outcomes 1 and 4 and relates to modules 1, 2, and 3.

Task Description

This assessment consists of a series of programming tasks, each requiring students to implement certain data structures and algorithms or apply common data structures and algorithms to solve well-defined problems.

These programming tasks are distributed across weeks 1, 2, and 3. The tutor supervises students to complete the coding parts of these tasks during each week's tutorial and workshop sessions, while students will complete the testing and reporting parts before the assessment submission due date. On-time completion of the coding parts is one of the marking criteria. By the end of each week's workshop session, the tutor checks each student's progress in completing the coding parts of the required tasks.

Task Instructions

The programming tasks cover data structures, sorting and searching algorithms. For each task, you need to write solution code to solve the problem, provide a set of inputs and outputs to test the solution, attach screenshots of the tests, and include a summary of comparison analysis if required.

If GenAI is used for the task, you must document how you use it and how it assists you in completing the tasks. Failing to do that will be subject to an academic integrity investigation.

Module 1: Basic data structures (coding to be completed during Week One's tutorial and workshop sessions)

Task 1 (2 marks): In HTML (Hypertext Markup Language), tags exist in both opening and closing forms and must be balanced to properly describe a web document. For example, the following simple HTML document shows the matching and nesting structure for tags in the language.

```
<html>
      <head>
            <title>
                Example
            </title>
      </head>

      <body>
            <h1> Hello, world </h1>
      </body>
</html>
```

Write a program that checks an HTML document for proper opening and closing tags. The input is an HTML file, and the output is whether this file contains unbalanced opening and closing tags. *Assume there is a space between each token within a line.*

Task 2 (3 marks): Design and implement an experiment to make benchmark comparisons of two queue implementations, one using *ArrayList* and the other using *LinkedList*. Construct two large queues (at least 100,000 elements) using the two implementations, respectively. Compare their *enqueue* and *dequeue* performance and analyse the results.

Hint: You can use the following code template to obtain the execution time:

```
long startTime = System.currentTimeMillis();
perform the task;
long endTime = System.currentTimeMillis();
long executionTime = endTime - startTime;
```

Task 3 (3 marks): The lexicographic order is defined for tuples by: (1) If a > c, then (a,b) > (c,d), and (2) If a = c and b > d, then (a,b) > (c,d). For example, the lexicographical order (3,1) > (1,10), (8,11) > (8,9), and (2,5) < (5,2).

Write a program that repeatedly gets a tuple input (a, b), inserts it into a list while keeping the list sorted in ascending lexicographical order, and outputs all the tuples in the list. The program terminates when the input tuple is (-1,-1).

Module 2: Sorting algorithms (coding to be completed during Week Two's tutorial and workshop sessions)

Task 4 (2 marks): Write a program that inputs a series of numbers and returns the length of the longest consecutive sequence of numbers present. For example, if the input is 8 3 2 1 10 7 9 7 6 21, then the length of the longest sequence of consecutive integers is 4 (the longest sequence of

consecutive integers is 7 8 9 10). *Note: you only need to output the length of the longest sequence, not the longest sequence itself.*

Task 5 (3 marks): The performance of quicksort depends on the pivot value. The quicksort algorithm presented in the module selects the first element in the list as the pivot. Revise it by selecting the median among the first, middle, and the last elements in the list. Design and run an experiment of sorting 100,000 elements to compare the two implementations and analyse the results.

Hint: You can use the following code template to obtain the execution time:

```
long startTime = System.currentTimeMillis();
perform the task;
long endTime = System.currentTimeMillis();
long executionTime = endTime - startTime;
```

Task 6 (3 marks): Sets are special since their elements characterise them. For example, if there is a repeating element, such as {1, 2, 3, 1}, you should simplify it to {1, 2, 3}. Write a program that takes two arrays, X and Y, as input and outputs an array Z that stores the intersection of X and Y, i.e., Z = X ∩ Y. For example, if X = [4, 1, 3], Y = [3, 6, 1], then the program should return Z = [1, 3]. Your solution MUST NOT use the brute force approach, i.e., avoid making all possible comparisons to guarantee your program works for large sets. Design an experiment to compare your solution against the brute force approach and analyse the results.

Hint: You can use the following code template to obtain the execution time:

```
long startTime = System.currentTimeMillis();
perform the task;
long endTime = System.currentTimeMillis();
long executionTime = endTime - startTime;
```

Module 3: Searching algorithms (coding to be completed during Week Three's tutorial and workshop sessions)

Task 7 (2 marks): Recall in the module that the binary search doesn't always return the first occurrence of the target value. But it provides a pointer close to the desired value. You can find the first occurrence of the target by traversing the structure back until no more occurrences of the target value are found. Modify the binary search algorithm to return the target value's first occurrence.

Task 8 (3 marks): Given a sorted list X, write a program that inserts a new element according to the following conditions: (1) If the element is present in X, then your solution must insert the new value after all the other occurrences with the same key as the new value, (2) If the element is not present in X, it should be inserted so that the list X continues to be sorted.

Resources

Use the following resources to support you when working on this assessment.

- Unit Modules 1-3 on Blackboard
- The reference books

- Java API documentation https://docs.oracle.com/en/java/javase/21/docs/api/index.html

Referencing Style Resource

Use Harvard style where applicable. Link: https://libguides.scu.edu.au/harvard

Task Submission

Submission of this assessment consists of two parts:

1. Coding parts: Completed during the respective tutorial and workshop sessions and checked by the tutor.
2. Portfolio: Submitted to the Blackboard unit site by the due date.

The portfolio must use the provided template, which includes code, testing inputs and outputs, screenshots of tests, answers to questions, analysis of experiment results, and reflection on the use of online resources, including GenAI, if any, for assisting in completing the tasks.

When you complete the assignment, you are required to submit the portfolio in Word DOCX file format using the **Turnitin** submission portal titled *Assessment 1 – Programming tasks: applications of data structures and algorithms* in the **Assessments Tasks and Submission** section on the Blackboard COMP6008 site. Only Microsoft Word documents submitted via the Turnitin portal on Blackboard will be accepted. The file will be named using the following convention: *LastNameFirstNameInitial_COMP6008_A1.docx, e.g., if a student's name is John Smith, the file name would be SmithJ_COMP6008_A1.docx*. ***Note: the code must be in text format and have appropriate indentation styles and comments. Screenshots of code are not accepted.***

Resubmit policy: This assessment is not eligible for a re-submit.

Academic Integrity

At Southern Cross University, academic integrity means behaving with the values of honesty, fairness, trustworthiness, courage, responsibility and respect in relation to academic work.

The Southern Cross University Academic Integrity Framework aims to develop a holistic, systematic and consistent approach to addressing academic integrity across the entire University. For more information, see: SCU Academic Integrity Framework

NOTE: **Academic Integrity breaches include** unacceptable use of generative artificial intelligence (GenAI) tools, the use of GenAI has not been appropriately acknowledged or is beyond the acceptable limit as defined in the Assessment, poor referencing, not identifying direct quotations correctly, close paraphrasing, plagiarism, recycling, misrepresentation, collusion, cheating, contract cheating, fabricating information.

At SCU the use of GenAI tools is acceptable, *unless it is beyond the acceptable limit as defined in the Assessment Item by the Unit Assessor*.

GenAI May be Used

Generative artificial intelligence (GenAI) tools, such as ChatGPT, **may be used** for this assessment task. If you use GenAI tools, you must use these ethically and acknowledge their use. To find out how to reference GenAI in your work, consult the referencing style for your unit via the Library referencing guides. If you are not sure how to, or how much, you can use GenAI tools in your studies, contact your Unit Assessor. If you use GenAI tools without acknowledgment, it may result in

an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

You **may use** Generative Artificial Intelligence (GenAI) tools, such as ChatGPT or Copilot, for this assessment task **to get some ideas or insight about particular problems in code**. It is similar when you try to find a snippet of code for doing a particular task in Stack Overflow. For example, **you must make your own effort to modify, refine, or improve the code to solve the assessment problems**. Think of it as a tool – a quick way to access information – or a (free) tutor to answer your questions. However, just as if you Googled something, you still need to evaluate the information to determine its accuracy and relevance. ***If you have used a GenAI tool in this assessment, you must document how you used it and how it assisted you in completing your assessment tasks***. ***Failing to do that will be subject to an academic integrity investigation***.

You **cannot use AI to generate a complete solution code or write a report**. You need to put your own effort into building the solution code for the assessments to demonstrate the required skills. Refer to assessment information in the Assessment Tasks and Submission area for details.

Special Consideration

Please refer to the Special Consideration section of Policy.
https://policies.scu.edu.au/document/view-current.php?id=140

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of Policy.
https://policies.scu.edu.au/view.current.php?id=00255

Grades & Feedback

Assessments that have been submitted by the due date will receive an SCU grade. Grades and feedback will be posted to the 'Grades and Feedback' section on the Blackboard unit site. Please allow 7 days for marks to be posted.

… continued on next page ...

Assessment Rubric

Note: The number of criteria used will vary depending on the complexity of the assessment task. Typically, 4–5 criteria are used.

Marking Criteria and % allocation	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail 0–49%
Criterion 1: Quality of solutions **ULOs**: 1, 4 **Percentage**: 40%	The solutions completely and accurately solve the problems as specified.	The solutions mostly solve the problems as specified.	The solutions seem to solve the problems as specified.	The solutions seem to partially solve the problems as specified.	No or few solutions are provided.
Criterion 2: Testing of solutions **ULOs**: 1, 4 **Percentage**: 20%	The solutions are thoroughly validated by tests.	The solutions are largely validated by tests.	The solutions are half validated by tests.	The solutions are sparsely validated by tests.	No or few tests are conducted.
Criterion 3: Design and analysis of experiments **ULOs**: 1, 4 **Percentage**: 10%	The experiments are solidly designed, and the results are thoroughly analysed.	The experiments are reasonably designed, and the results are reasonably analysed.	The experiments are designed, but the results are not well analysed.	The experiments are partially designed, and the results are incomplete.	No or few experiments are designed.
Criterion 4: On-time completion of solution codes **ULOs**: 1, 4 **Percentage**: 30%	Complete all the required coding parts during the respective tutorial and workshop sessions.	Complete 2/3 of the required coding parts during the respective tutorial and workshop sessions.	Complete half of the required coding parts during the respective tutorial and workshop sessions.	Complete 1/3 of the required coding parts during the respective tutorial and workshop sessions.	Fail to complete any required coding part during the respective tutorial and workshop sessions.

Description of SCU Grades

High Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows exceptional ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as outstanding in relation to the learning requirements specified.

Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows a well-developed ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as distinguished in relation to the learning requirements specified.

Credit:

The student's performance, in addition to satisfying all of the basic learning requirements specified, demonstrates insight and ability in researching, analysing and applying relevant skills and concepts. The student's performance could be described as competent in relation to the learning requirements specified.

Pass:

The student's performance satisfies all of the basic learning requirements specified and provides a sound basis for proceeding to higher-level studies in the subject area. The student's performance could be described as satisfactory in relation to the learning requirements specified.

Fail:

The student's performance fails to satisfy the learning requirements specified.