PROG5001: FUNDAMENTALS OF PROGRAMMING

Summary

Title	Assessment 2
Type	Practical
Individual or Group	Individual
Due Date	Week 6 Monday (5 Aug) at 11:59 pm AEST
Length	N/A
Weighting	40%
Academic Integrity	Contract cheating and any use of GenAI, such as ChatGPT, in this assignment are strictly prohibited. Any breach may have severe consequences. Please read carefully the "Academic Integrity" section below.
Submission	You will need to submit the following to the submission link provided for this assignment on the MySCU site: • The link to your GitHub repository • The BlueJ project with all your Java code files • A text file with the pseudo code of algorithms 3 and 4
Unit Learning Outcomes	This assessment task maps to the following ULOs: ULO2: modify an existing program or code segment to achieve an intended outcome. ULO3: design or select suitable data structures and algorithms for a problem. ULO4: apply algorithmic thinking and programming techniques in solving simplified computational problems.

Rationale

The purpose of this assessment is to test your ability to:

- Analyse the execution of a program or code segment to determine the output of the code.
- Modify an existing program or code segment to achieve an intended outcome.
- Design or select suitable data structures and algorithms for a problem
- Apply algorithmic thinking and programming techniques in solving simplified computational problems.

through developing a simple program to manage student marks. Your work must demonstrate your learning over the first five modules of this unit. Real-world software developers conduct similar thinking process and analysis to develop efficient and effective algorithms.

Task Description

Your task in this assessment is to develop a program that calculate several statistics of students' marks. You will design a program that takes a series of students' marks from a file and conduct total, conditional filtering, and sorting. You need to develop the program from scratch for all functionalities, you cannot use any library functions.

Task Instructions

Your task is to develop a simple program to read student marks and filtering sorting them. The program will have the following functional requirements:

- **Read inputs from file (aligns with ULO2, ULO3):** Reads the unit name and students' marks from a given text file. The user will provide the file name. The file contains the unit name and the list of students with their names, student IDs and marks for three assignments. The file also contains lines, which are comments, and your program should check to ignore them when reading the students' marks.
- **Compute and print student-wise marks (aligns with ULO3):** Calculates the total mark for each student from the assessment marks and print out the list of students with their name, student ID, assessment marks and the total mark.
- **Filter and print students and their marks (aligns with ULO3, ULO4):** Prints the list of students with total marks less than a certain threshold. The threshold will be entered from the keyboard by the user. To complete this function requirement, you are required to perform two steps:
    - **Step 1:** You will create an algorithm (called Algorithm 3) in **pseudo code** to print the list of students with total marks less than a certain threshold. The algorithm will take the threshold as an input.
    - **Step 2:** You will implement Algorithm 3 based on the pseudo code developed in Step 1. Note that the implementation must be in line with the pseudo code, and you are **not allowed** to use existing searching and sorting library functions in Algorithm 3.
- **Sort marks and filter students (aligns with ULO3, ULO4):** Prints the top 5 students with the highest total marks and the top 5 students with the lowest total marks. You are required to perform two-step process to complete this function requirement:
    - **Step 1:** You will create an algorithm (called Algorithm 4) in **pseudo code** to print the top 5 students with the highest total marks and the top 5 students with the lowest total marks.
    - **Step 2:** You will implement Algorithm 4 based on the pseudo code developed in Step 1. Note that the implementation must be in line with the pseudo code, and you are **not allowed** to use existing searching and sorting library functions in Algorithm 4.
- **Create menu system (aligns with ULO2):** Create a simple menu system to allow users to select and execute each function. This does not require to create any graphical user interface.

You will also **create a repository on GitHub** to store your project work with all files and documents. You **must show your work progress** in this assignment by regularly committing your project to the GitHub repository. **Failing to show the correct work progress will fail the assignment.**

You will treat this task as a creative exercise. Think about innovative ways to develop and implement the Algorithm 3 and 4. You should use 'best 'practice' throughout, such as commenting on your code and spacing your code so it is readable, using reasonable naming, and validating your code to ensure it conforms to coding standards.

In completion of this task, you will demonstrate:

- Algorithmic thinking skill: creating an algorithm to solve a simple computing problem
- Basic programming skills: using variables and programming constructs to create a new program from an existing program, debugging a simple program and fixing errors.
- Design or select suitable data structures and algorithms for a problem.

Recommendations

Complete the following tasks:

- Step 1: Read the task description carefully to clearly understand what you need to do. If you have questions, use the Discussion Board to ask.
- Step 2: Read the marking rubric to understand how your work will be assessed. If you have questions, use the Discussion Board to ask.
- Step 3: Create a GitHub repository for the project.
- Step 4: Study modules 1 to 5 and spend 10 hours each week to complete the task. Commit your work to the GitHub repository regularly.

Resources

To complete the task, you are recommended to:

- Study modules 1-5 materials and complete all learning activities
- Take an active role in the weekly tutorial and workshop. Bonus marks are given to your contributions and will be considered in making decision about your final grade.

Task Submission

Submit the work to the provided submission link on the Blackboard learning site. You will submit:

- The link to your GitHub repository
- The project with all source files, along with the test file with student details and marks.
- A text file with the pseudo code of algorithms 3 and 4

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.

GenAI May Not be used for Assessment 2

Generative artificial intelligence (GenAI) tools, such as ChatGPT, **must not be used** for this assessment task. **You are required to demonstrate that you have developed the unit's skills and knowledge without any external support.** If you use GenAI tools in your assessment task, it will result in an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

Since you are mastering fundamental skills, you are permitted to work from the examples in the MySCU site or study guide, but you must acknowledge assistance from other textbooks, classmates, or online resources. In particular, you must not use online material or help from others, as this would prevent you from mastering these concepts.

This diagram will help you understand where you can get help:

Lecturer Tutors	GenAI e.g., ChatGPT	Online Forums
Relatives	Students outside unit	Hired coders
Classmates	Private Tutors	Other

Encouraged

Attribution Required (in the report)

Ask tutor

Not acceptable

Please note that if your marker has any suspicion that you had help with your work or is not your own you will be asked to come to a meeting with your marker to explain your work. Any student who is unable to explain their code will be submitted for academic misconduct.

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.

Assessment Rubric

Marking Criteria and % allocation	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail 0–49%
Design and implement file reading and printing (ULO2, ULO3) – 10%	The program meticulously reads the unit name and student data from the text file, accurately ignoring comment lines and handling all file errors gracefully. It accurately calculates total marks and prints a well-formatted list of students, demonstrating exceptional attention to detail and robustness. The code is thoroughly documented and adheres to best practices.	The program correctly reads the unit name and student data from the file, ignoring comment lines and handling common errors adequately. It correctly calculates total marks and prints an organised list of students. The code handles most data inputs accurately, is well-documented.	The program reads the unit name and student data correctly, with minor issues in handling comment lines and some file errors. It calculates total marks and prints a list, with minor formatting issues. The code follows basic practices and is moderately documented.	The program reads the unit name and student data but inconsistently handles comment lines and file errors. It calculates total marks and prints a list, though the output may be inconsistently formatted. It performs basic calculations but struggles with complex inputs, with minimal documentation.	The program fails to address the requirements, lacking error handling and demonstrating poor efficiency. It fails to calculate total marks accurately or print a correct list of students. The code is insufficiently documented, does not follow standard practices.
Develop algorithm to filter and print (ULO3, ULO4) – 30%	The pseudo code is exceptionally clear, detailed, and logically precise. The implementation is precise, adhering closely to the pseudo code, and demonstrates exceptional coding practices, avoiding searching/sorting libraries. The code robustly handles the	The pseudo code is well-structured and logically structured. The implementation closely follows the pseudo code and demonstrates good coding practices, without using searching/sorting libraries. The code effectively handles the input threshold and student filtering and printing.	The pseudo code is logically correct. The implementation mostly follows the pseudo code and avoids using searching/sorting libraries. The code correctly handles the input threshold and student filtering and printing, with some minor limitations.	The pseudo code handles the requirements in some extent but contains logical gaps or inefficiencies. The implementation somehow follows the pseudo code with moderate inconsistency, and does not use searching/sorting libraries. The code performs basic	The pseudo code is incomplete, illogical, or fails to address the requirements. The implementation deviates significantly from the pseudo code or uses searching/sorting libraries. The code fails to filter and print student information.

	input threshold and student filtering and printing.			thresholding and student filtering and printing, but may produce some errors.	
Develop algorithm to sort and filter (ULO3, ULO4) – 40%	The pseudo code exceptionally clear, detailed, and logically structured, effectively sorting marks and filtering students. The implementation is precise, strictly adhering to the pseudo code, and demonstrates exceptional coding practices without using searching/sorting libraries. The code is highly efficient and robust, accurately sorting and printing the student lists.	The pseudo code is well-structured and logically sound, efficiently sorting marks and filtering students. The implementation closely follows the pseudo code and demonstrates good coding practices, avoiding searching/sorting libraries. The code is efficient and correctly sorts and prints the student lists.	The pseudo code is logically correct to sort marks and filter students. The implementation mostly follows the pseudo code and avoids using searching/sorting libraries. The code correctly sorts and prints the student lists, though there may be minor limitations.	The pseudo handles the requirements in some extent but contains logical gaps or inefficiencies. The implementation follows the pseudo code but may deviate slightly, and does not use searching/sorting libraries. The code performs basic sorting and printing of the student lists but may produce errors.	The pseudo code is incomplete, illogical, or fails to address the requirements. The implementation deviates significantly from the pseudo code or uses searching/sorting libraries. The code fails to sort and print student and mark information.
Design and develop a menu system (ULO2) – 10%	The menu system is exceptionally well-designed, providing a clear and intuitive text-based interface that allows users to select and execute each function seamlessly. It includes robust input validation and error handling, ensuring that the system is user-friendly and	The menu system is well-constructed, offering a clear and functional text-based interface that allows users to select and execute each function effectively. It includes adequate input validation and error handling, making the system user-friendly and robust against most invalid	The menu system provides a functional text-based interface that allows users to select and execute each function. It includes basic input validation and error handling, making the system generally user-friendly. The code is reasonably efficient and moderately documented,	The menu system offers a basic text-based interface that allows users to select and execute each function, though it may lack clarity or user-friendliness. Input validation and error handling are minimal. The code is functional but may cause errors.	The menu system is poorly designed, providing an unclear or dysfunctional text-based interface. Input validation and error handling are inadequate or non-existent. The code is inefficient and does not follow standard practices.

	resilient to invalid inputs. The code is highly efficient and follows best coding practices.	inputs. The code is effective and following good coding practices.	adhering to basic coding practices.		
Apply best practices (ULO4) – 10%	Demonstrated an exceptional level of application of coding best practices throughout the program. Repository on GitHub was created, thoroughly documented and professionally maintained.	Demonstrated a proficient level of application of coding best practices. Repository on GitHub was created, documented in detailed manner and properly maintained.	Demonstrated an adequate level of application of coding best practices. Repository on GitHub was created, documented and maintained with acceptable standard.	Demonstrate the basic level of application of coding best practices. Repository on GitHub was created, documented and maintained with some limitations.	Demonstrate a poor level of application of coding best practices in the program. Repository on GitHub was not created or was not documented and maintained as an acceptable standard.

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.