INTI International College Penang **School of Computing**

3+0 Bachelor of Science (Hons) in Computing, in collaboration with Coventry University, UK

Coursework Cover Sheet

Section A - To be completed by the student

Full Name:	
CU Student ID Number:	
Semester:	
Lecturer: **Koo Lee Chun**	
Module Code and Title: **5003CEM Advanced Algorithms**	
Assignment No. / Title: **Individual Report**	% of Module Mark **67%**
Hand out date: **20 APRIL 2023**	Due date: **9 JUNE 2023**
Penalties: No late work will be accepted. If you are unable to submit coursework on time due to extenuating circumstances you may be eligible for an extension. Please consult the lecturer.	
Declaration: I the undersigned confirm that I have read and agree to abide by the University regulations on plagiarism and cheating and Faculty coursework policies and procedures. I confirm that this piece of work is my own. I consent to appropriate storage of my work for plagiarism checking. Signature(s): ----------	

Section B - To be completed by the module leader

Intended learning outcomes assessed by this work:

LO1: Understand and select appropriate algorithms for solving a range of problems and reason about their complexity and efficiency.
LO2: Design and implement algorithms and data structures for novel problems.
LO3: Understand the intractability of certain problems and implement approaches to estimate the solution to intractable problems.
LO4: Describe the issue of data consistency in non-synchronous applications.
LO5. Design and implement a basic concurrent application.

Marking scheme	Max	Mark
Refer to rubric at the end of the page		
Total		

Lecturer’s Feedback
Internal Moderator’s Feedback

Instructions:

1. There will be **FOUR(4)**programs to be completed and submit before the due date. The programs will cover a range of data structure or algorithms taught on the module, as well a basic concurrent application.

   You are required to submit the commented source code (Pycharm or dev c++ project) of your solutions in zip format.

2. Prepare a documentation (in pdf or Microsoft word) for the programming tasks and submit before the due date. The documentation should include:
   - Screen capture for each program output and explanation. Do include your discussion here too for question that asks for it.
   - The weakness of your solution or/and what is not working in your solutions
   - Reflection – challenges and what you had learnt from this coursework.
   - References – list of all the references that you reference to in this assignment including code reference too.
   - Appendix - List of your solutions (No screen capture is acceptable).

3. There will be a private VIVA session for each member in the team to demonstrate your understanding and originality of your work. You are required to explain how your code works. Instructor will ask you a structured set of questions about your code.

4. Read the important notes and marking rubric at the end of this documents to avoid any marks penalty.

Programming Tasks

Question 1: Cake ordering System

Design and develop a Cake ordering system based on following requirements:

- The program should be menu driven, giving the user various choices of operations that allows a user to place an order for the cake(s), view list of cake order details, modify or delete a particular order if necessary.

- For every cake order, the following information willl be stored:
  - Order ID (Auto assigned, no duplication), Cake Code, flavor (eg: Strawberry, chocolate), weight (Kg), Unit Price, Qty and customer information who order the cake.
  - The customer information consists of customer ID, name, address and contact number.
  - The system shall display additional information that is amount (unit price * qty) when viewing the order details.
- The program must use BST data structure to facilitate each operation.
- The system shall demonstrate a good OOP design, data validation and error handling.

Question 2: Hash Table Collision technique average cost comparison

Write a program to compute and compare two collision techniques: Chaining and open addressing (linear probing) for the hash table implementations.

The program shall insert 5,000 randomly generated numbers into two separate hash table objects that use different methods (chaining and open addressing) to handle the collision. Use 6001 as the table size of the hash tables.

The program shall execute at least 10 times and calculate the maximum, minimum and average number of collisions that had happened for each technique and run. Populate the result in the following tabular formats:

Average Cost					
Chaning			Open Addressing		
Minimum	Maximum	Average	Minimum	Maximum	Average

Note: reminder to clean up the hash table before the next run.

Discuss your observation and finding.

Question 3: Graph

Write a program that uses graph concept. The program shall fulfil the following requirements:

- construct a weighted directed class graph that consists of the following methods:
  - listAdjancentVertex : List all the adjancent vertex for a given vertex
  - sumHighestAdjacentVertex : sum up all the weight for all the adjacent Vertices.
- Create two different graph objects (least 6 vertexes for each graph) and call the above two functions and display the results.

Explain the method/design you had used to present the graph structure in your program.

Question 4: Concurrent process

(a) Write a function to calculate and display the car loan monthly repayment. Assume flat interest rate is used.

Refer to the link below to understand how to calculate the monthly repayment of a loan:
https://www.comparehero.my/personal-loan/articles/heres-how-car-loans-work-and-why-interest-charges-are-higher-than-you-think

(b) Write a program to allow users to calculate monthly repayment for 3 customers. The program shall call the function define in (a) concurrently.

Observe the output of your solution. Discuss the theorical behind to support your observation in the documentation.

Important notes:

- Label your question number clearly in your documentation
- In Appendix session, No print screen of codes from your programming IDE is allowed. No mark will be rewarded to you if your codes are in screenshot view and could not be checked for plagiarism. Any suspect plagiarism will be strictly follow academic disciplinary board procedure.
- You will get penalty if your similarity index is more than 30%.
- Submit your works in the canvas before the deadline. **Late submission** will be awarded **zero mark**.
- Viva is required to assess your understanding of your solutions and originality of your work. No attending VIVA session will be awarded **zero mark.**

Marking scheme:

	CODE SUBMISSION 40 MARKS	**Documentation 10 marks**	**VIVA 50 MARKS**		
Marking criterion	Code 40	Documentation 10	Knowledge of code 30	Critical reflection 10	Clarity of communication 10
1st (70+)	Code is fully working, bug-free, well-commented, possibly with some advanced features.	Each section in the documentation are organised with relevant title/sub title. Discussions are very comphrehesive and in depth.	Student shows sophisticated and detailed knowledge of how the code works at every level. Student can explain the design and implementation decisions.	Able to critique the implementation and offer a range of alternative possibilities; will be able to explain implementation decisions at a sophisticated level, including complexity analysis.	Able to speak clearly and coherently in a well-prepared way, and without the need for frequent prompting. Authoritative on the material. Great question-handling.
2.1 (60-69)	Code works, may have one or two non-important bugs. Commented. Does not has any advanced features.	Each section in the docuemntation are mostly organised with mostly relevant title/sub title. Discussions are somehow comphrensive with very little areas that lack of details explanation	Good knowledge of how major sections of the code work. Is able to explain design decisions and why things were implemented in particular ways.	Able to critique the implementation against possible alternatives (which may be limited to one). Can broadly explain implementation decisions, but less effective on detail. May be less effective on complexity analysis.	Generally able to speak clearly and coherently without the need for many prompts. Clear ability to communicate what's been done. Most questions well handled.

2.2 (50-59)	Code generally works, may have bugs, and may not cover all the required features even at basic level. Has some comments.	Each section in the docuemntation are organised with mostly relevant title/sub title. Some discussions are moderate with some discussion are in depth and some lack of details.	Some knowledge of how the code works but there may be areas of confusion. Ability to explain basic decisions about the design and implementation but again there may be misconceptions.	Less able to be critical about the code, and may not be able to discuss alternative implementations except in broad terms. May show confusion on complexity analysis.	Able to say something and can answer questions but may be hesitant. May need some prompting, but insight shown. Some questions may not be well answered and some answers can be wrong.
3rd (40-49)	Code may not work; may not cover basic required features, may have significant bugs; but shows potential to work. May have some comments.	Each section in the docuemntation are mostly organised with some relevant title/sub title, with some topic not included. A few discussions are comprehesive, but some areas lack of details.	Limited knowledge of how the code works which is unlikely to show insight at detailed levels. Aware of the design of the code but not much insight into different design decisions that could be made.	Unlikely to be critical about the code beyond describing what it does. Little awareness of possible alternatives or complexity.	Halting, hesitant. May show some elementary insight but unable to speak authoritatively. May be able to deal with a few questions but struggle with most.
Fail (0-39)	Code is incomplete, does not work, is not commented, does not show clear potential. At the zero end, nothing has been done.	Documentation is very brief with many section incomplete/no deliver. At the zero end, nothing has been done.	Little if any ability to explain how the code works. May be attempting to explain by reading through code and trying to work it out during the viva, where the results are largely or totally wrong / incomplete. At the zero level the explanation (if any) will have no value.	Largely unable to be critical about the code or implementation. At the zero end, student will be completely unable to evaluate the code.	Little knowledge or engagement with the module and the performance shows this. May be attempting to 'make up' answers on the spot. At the zero end, student may appear to be unaware of the problems set.