INTI International College Penang **School of Engineering and Technology**

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

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: **Courswork**	% of Module Mark **67%**
Hand out date: **28 April 2022**	Due date: **3 July 2022**
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:

LO2: Design and implement algorithms and data structures for novel problems.
LO3: Specify and implement methods to estimate solutions to intractable problems
LO5: Design and implement a basic concurrent application

Marking scheme	Max	Mark
Total	66	

Lecturer’s Feedback
Internal Moderator’s Feedback

Instructions:

1. There will be FOUR 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 (in Pycharm project format) of your solutions.

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 references to in this assignment including code reference too.
   - Appendix - List of your solutions.

3. There will be a private VIVA session for you 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: Tasks management

Write a program that uses hash table to store a list of tasks. The program shall fulfill the following requirements:

- Create a class hash table. Each task consists of task id and description. The class shall provides the following methods:
  - Insert a task into the hash table. Apply appropriate collision handling technique in the method
  - Retrieve a task from the hash table
  - Display hash table

- Provide a menu that allows a user to perform the following operations until the option 3 (exit) is selected:

  1) Add a new task
  2) Retrieve a task
  3) Exit

- If user select option 1 (Add new task), the program shall add a new task into a hash table object.

- If user select option 2(Retrieve a task), the program shall retrieve the task from the hash table. Display error message if the task is not found.

Question 2: AVL Tree and Binary Search Tree

Write a program to compare the runtime taken to search a number in typical binary search tree and an AVL tree. The program must fulfill the following:

- Define and implement a typical binary search tree class that handle insert and searching an integer value.
- Define and implement an AVL class that handle insert and searching an integer value.

- Generate n numbers randomly (1-1000). The n value shall be determined by the user. Use the generated ramdom number to prepare two datasets: random order list and a sorted list. Then insert those data into two separate Binary search tree. Do the same for AVL tree as well.

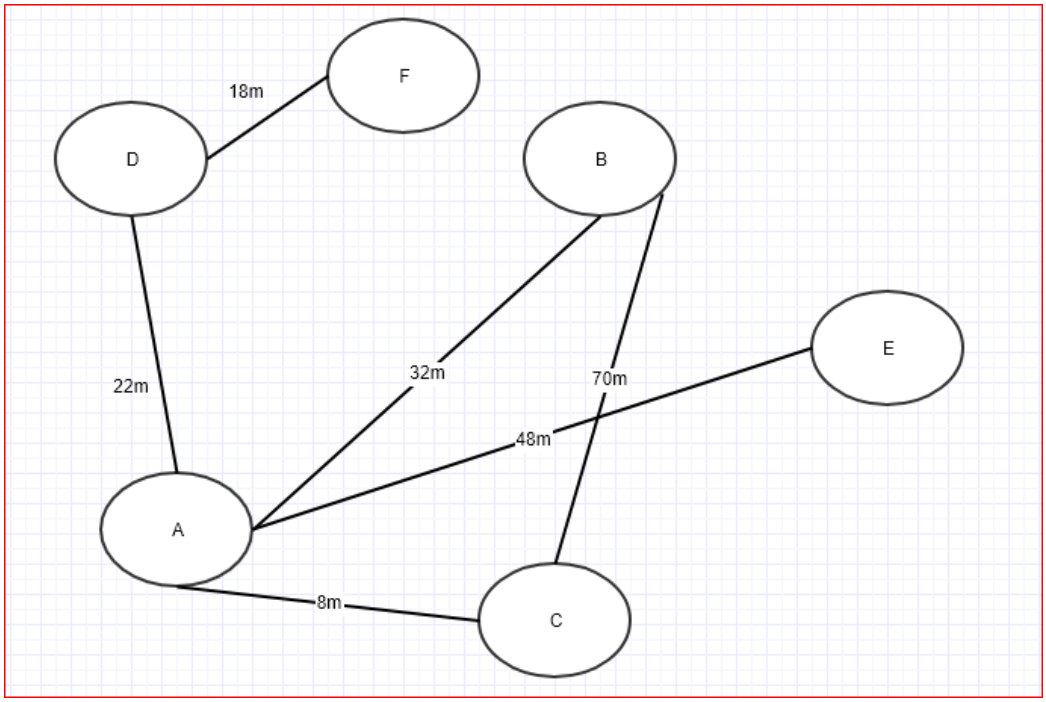
- Perform a search on a number 2000 (a number that never exist in any of BST and AVL)

- Display a summary report to compare the time time taken
  Note: you should have 4 runtime results.

In your documentation, discuss your observation in term of runtime. Compare and discuss your result with the theorical time complexity.

Question 3: Graph

The distances between the places in a small town are shown below:



n

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

- Construct a graph based on the above diagram.
- Ask the user to input 2 places to visit.
- Call a function to determine if the two places are next to each other. If there are next to each other, the function shall display their distance and return true, otherwise return false.
- If both places are not next to each other, call another function to determine if two places are reachable via other places. Display appropriate message to indicate the result.

In the documentation, discuss your approach or any specially theoritical algorithm you had used to solve this problem.

Question 4: Concurrent process

Define a functions that received 3 arguments: two integers value and an operation symbol (+, -, *, or /). The function shall returns the result after performing these operation.

Write a program that asks user to enter 2 numbers. Then, the program shall performs the operation of sum, substraction, product and division of these two numbers concurrently. For example, if user enter 10 and 2, then program shall display the result as shown in the following format:

10 + 2 = 12
10 / 2 = 5
10 – 2 = 8
10 * 2 = 20

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 documentations
- 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%.
- Upload to the Blackboard (**Final Submission**) 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-	Each section in the docuemntation	Good knowledge of how major	Able to critique the implementation	Generally able to speak clearly and

	important bugs. Commented. Does not has any advanced features.	are mostly organised with mostly relevant title/sub title. Discussions are somehow comphrensive with very little areas that lack of details explanation	sections of the code work. Is able to explain design decisions and why things were implemented in particular ways.	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.	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,	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

			where the results are largely or totally wrong / incomplete. At the zero level the explanation (if any) will have no value.		the spot. At the zero end, student may appear to be unaware of the problems set.