SC1003 – Introduction to Computational Thinking and Programming

Academic Year	AY2021	**Semester**	1	
Course Code	SC1003			
Course Title	Introduction to Computational Thinking and Programming			
Pre-requisites	NIL			
Pre-requisite for	SC1007, SC2002			
No of AUs	3			
Contact Hours	Lectures	13 with LAMS 26 without LAMS	Example Class (Seminars/Hands-on Exercises)	26

Course Aims

Computational thinking (CT) is the process of analysing a problem then designing and expressing its solution in such a way that a computer can effectively carry it out. It includes a number of characteristics, such as breaking a problem into small and repetitive ordered steps, logically ordering and analyzing data and creating solutions that can be effectively implemented as programs running on computer.

The aim of this course is hence to take students with no prior experience of thinking in a computational manner to a point where you can derive simple algorithms and code the programs to solve some basic problems in your domain of studies. Student will also learn about basic program construct and simple data structures. In addition, the course will include topics to appreciate the internal operations of a processor.

Intended Learning Outcomes (ILO)

Upon the successful completion of this course, you shall be able to:

1. Describe the internal operation of a basic processor, how a program is executed by a computer and computing trends.
2. Analyse a problem then design and express its solution in such a way that a computer can effectively carry it out. (i.e. equip you with CT skills)
3. Implement problem solutions as programs using basic control structures (sequence, conditional, iterative).
4. Implement problem solutions as programs using basic data types and aggregate data types.
5. Apply the CT concepts on case studies/problem-based scenarios through hands-on practice of the CT processes.

Course Contents

	Topics
0	**Course Overview and Concepts of Computational Thinking** Solving complex problem using computer - enables the student to work out exactly what to tell the computer to do.
1	**Overview of Programming Languages and Basic Internal Operation of Computer** High level programming languages (Python, C, Java) Basic computer organization (Processor, Memory, I/O) and how a computer execute a program (Machine instructions).
2	**Basic Program Structure: Control Constructs and Data Types** Concepts of data types, variables; Pseudo-code and flowcharts; Sequences, Selection (if/else), iteration (for/while loop).
3	**CT Concept - Abstraction** Problem formulation - reducing something to a very simple set of characteristics to only focusing on the most relevant to the problem. Concept of functions/libraries and data structure.
4	**CT Concept - Decomposition** Break a complex problem into smaller and more manageable parts/steps, such that each of these smaller problems can then be looked at individually.
5	**CT Concept – Pattern recognition** Looking for similarities among and within problems, which also enable re-use knowledge of previous similar problems.
6	**CT Concept – Algorithm** Reformulating the problem into series of ordered steps through Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. (Some common/useful examples: Sorting and searching).
7	Basic Programming Constructs in C Language C program structure. Syntax and semantics. Intrinsic data types, declarations, operators, assignments, control flow, and simple input/output; Pre-processing. Functions; Return values, arguments and parameter passing; Scopes of variables; Concept of side effects.
8	Built-in Data Structures Pointers, pointer operations and pass by reference; One-dimensional and multi-dimensional arrays, and pointers and arrays; Character strings and arrays of strings; Structures, arrays of structures and type definitions.

Assessment (includes both continuous and summative assessment) CE/CZ1103

Component	Course LO Tested	Related Programme LO or Graduate Attributes	Weightage	Team/ Individual	Assessment Rubrics
1. TEL MCQs	1,2,3,4,5	a, b, c, d	10%	Individual	See appendix 1
2. Online MCQs based quizzes	2,3,4,5	a, b, c	20%	Individual	See appendix 1
3. Hands-on exercises completion and assessment	2,3,4,5	a, b, c, d, e, j, l	20%	Individual	See appendix 2
4. Assignment	2,3,4,5	a, b, c, d, h, l	15%	individual	See appendix 3
5. Test	2,3,4,5	a, b, c, d, h, l	35%	Individual	See appendix 3
Total			100%		

Appendix 1: Assessment for Online tasks

You will take 2 MCQs based quizzes. The maximum score is 20.
You will complete 12 online video with MCQs. The maximum score is 10.

Appendix 2: Assessment for Hands-on programming exercise in Python

You are required to show working program to lab supervisor in each lab session. The maximum score is 5.
You will take 1 MCQs based quiz related to the programming exercises done in the class. The maximum score is 15.

Appendix 3: Assessment Criteria for Assignments and Tests in C

Assignments and lab tests involve the submission of programming code for grading. In this course, the submitted code will be marked automatically by an online programming submission and grading system. First, programming questions related to LO 3 to LO 4 from lab assignments and lab tests will be set and given to individual students. Then, you will submit your answer code within a specified duration. Finally, the submitted code will be marked automatically by the online system.

For grading, instead of marking according to the similarity between the submitted answer code and the solution code, we focus on marking based on the correctness of the programming logic to solve the given problem. Questions will be designed carefully to test the different concepts in each topic of the course. In addition, it is also important to note that programs will only be meaningful if they can run correctly. The marking will be carried out using the automated marking mechanism in the online system according to test cases. As such, the assessment rubrics are given as follows.

Score	Interpretation
5	Correct code – The submitted code is able to compile and run correctly on all the test cases
1-4	Partially correct code – The submitted code is able to compile and run correctly on some test cases. The score will be allocated proportionally according to the number of test cases which can be run correctly.
0	Incorrect code or non-compilable code - The submitted code is unable to run correctly on all the test cases, or the submitted code is unable to compile.

An Example:

- Problem: Write a C program that reads the user input on temperature in degrees Fahrenheit, and then converts the temperature from degrees Fahrenheit into degrees Celsius. The relevant formula is given as follows: Celsius = (5/9)*(Fahrenheit – 32).
- There are two test cases defined for this program:
  (1) Test Case 1:
  ```
  Enter the temperature in degree F:
  45
  Converted degree in C: 7.22
  ```
  (2) Test Case 2:
  ```
  Enter the temperature in degree F:
  -12
  Converted degree in C: -24.44
  ```

- Correct code – if the submitted code is able to compile and run correctly against all the test cases, a score of 5 will be awarded.
- Partially correct code – if the submitted code is able to compile and run correctly on the first test case, but fails to run on the second test case, or vice versa, a score of 2.5 will be awarded.
- Incorrect code – if the submitted code is unable to run correctly on both test cases, a score of 0 will be awarded.
- Non-compilable code – if the submitted code is unable to compile (e.g. due to syntax errors), a score of 0 will be awarded.

Note: If you were found cheating or plagiarism in assignments or lab tests, your score for that assignment or lab test will be zero. In addition, you will also be subjected to disciplinary action according to University regulations.