SC1003 Freshmen Exemption Test Information

Syllabus

Computational thinking (CT) is the mental skills and practices for solving problems and designing computations. Computations are complex series of numerical calculations and symbolic manipulations which a computer follows to do jobs for us.

The aim of this course is hence to have students master the CT methods where you are able to analyse a problem then design and express its solution in such a way that a computer can effectively carry it out. It includes a number of characteristics, such as breaking a complex problem into a number of smaller/simpler problems, logically ordering numerical calculations and symbolic manipulations to create solutions that can be effectively implemented as programs, running on a computer. Student will also learn a programming language. In addition, the course will include topics to appreciate the internal operations of a processor.

Course Overview and Concepts of Computational Thinking What computational thinking is; How is computational thinking used; What is computational thinking not.	*Computational Thinking Concepts*
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).	*Machine language and high-level programming language* *Binary and Hexadecimal Numbers, ALU operation*
Basic Program Structure: Control Constructs and Data Types Concepts of data types, variables; Sequences, Selection (if/else), iteration (for/while loop).	*Pseudo code and flowchart; Data type and Variables* *Boolean relational Operators, selection* *Repetition*
CT Concept – Abstract Problem formulation - reducing something to a very simple set of characteristics to only focusing on the most relevant to the problem. Data abstraction, procedural abstraction, concept of functions/libraries.	*Data abstraction (Data structure)* *Procedural abstraction (function development)*
CT Concept - Decomposition Problem decomposition: break a complex problem into smaller and more manageable parts; task decomposition: break a complex task into smaller and more manageable subtasks; Repeated decomposition till each of these smaller	*Divide and Conquer* *Recursion* *Case studies*

problems/tasks can then be looked at individually; Pseudo-code and flowcharts.	
CT Concept – Pattern recognition Looking for similarities among and within problems, which also enable re-use knowledge of previous similar problems.	
CT Concept – Algorithm What is an algorithm. The importance of a clear and efficient algorithm.	*Sorting algorithms* *Searching algorithms*

Reference Book

Think Python (Online Version) Chapters 1 to 13: **https://greenteapress.com/wp/think-python-3rd-edition/**

Sample Questions

1. Reimplement a version of the built-in *len* function in Python for computing the length of a list. Write three versions:

    1. A version that is recursive but does not use an accumulator.
    2. A version that uses an accumulator and is tail recursive.
    3. A version using iteration (i.e., no recursion).

2. The *Hamming weight* of a binary string is the number of 1s occurring in it. Write a recursive Python function that takes in integers $n \geq 1$, $c \geq 0$ and returns the list of all binary strings (in any order) of length n and Hamming weight at most c.

3. Given an example of a list (or more generally, class of lists) for which bubble sort attains its *best-case* runtime. Then give a list (resp., class of lists) for which it attains its *worst-case* runtime.