**SECTION 2/3 Check Sheet 2**

**Ch 7,8,9,10,11**

|  |  |
| --- | --- |
| **Topic**  **Highlight as Required. I understand and know this topic fully, partially, not at all** | **Comments/Actions required** |
| **1.1 Data Structures**   * Be familiar with the concept of data structures   (KC: an organised collection of related data which has special rules for accessing and changing the data) |  |
| **1.2. Single and Multi-dimensional arrays**   * Use arrays or equivalent in the design of solutions to simple problems. * A 1D array is useful to represent a vector. (KC a single column. Eg in python a list of names) * A 2D aray is useful to represent a matrix. (KC a table. Eg in python a list of lists – eg ten test marks for all the student in a class of 20 students) * More generally an n dimensional array is a set of elements of the **same data type** that are indexed by a tuple of n integers where a tuple is an ordered list of elements.   (KC a multi-dimensional table. Eg in python a list of list of lists…… etc. Eg ten test marks for 3 different subjects for 20 students who all happen to do the same three subjects) |  |
| **1.3 Fields, records and files**   * KC: Make sure know the difference between a field, a record and a file * Be able to read/write from/to a text file * Be able to read/write from/to a binary file |  |
| **1.4 Abstract data types/data structures**   * Be familiar with:   + Concept of a queue   + Uses of a queue   + Concept of a stack   + Uses of a stack   + Concept of a graph   + Uses of a graph   + Concept of a tree   + Uses of a tree   + Concept of a hash table   + Uses of a hash table   + Concept of a dictionary   + Uses of a dictionary   + Concept of a vector   + Uses of a vector   + How you might represent a queue in python   + How you might represent a stack in python   + How you might represent a graph in python   + How you might represent a tree in python   + How you might represent a hash table in python   + How you might represent a dictionary in python   + How you might represent a vector in python * Distinguish between static and dynamic data structures * Compare uses of static data structures and dynamic data structures * Explain the advantages and disadvantages of static and dynamic data structures. |  |
| **2.1 Queues**   * Be able to describe and apply the following **to linear queues**, **circular queues** and **priority queues:**   + Add an item   + Remove an item   + Test for an empty queue   + Test for a full queue |  |
| **3.1 Stacks**   * Be able to describe and apply the following operations:   + Push   + Pop   + Peek or top (returns the value of the top element **without** removing it)   + Test for empty stack   + Test for stack full |  |
| **2.4 Graphs**   * Be aware of a graph as a data structure used to represent more complex relationship. * Be familiar with typical uses for graphs * Be able to explain the terms:   + Graph   + Weighted graph   + Vertex/node   + Edge/arc   + undirected graph   + directed graph * Know how an adjacency matrix may be used to represent a graph * Know how an adjacency list may be used to represent a graph * Be able to compare the use of the adjacency list with the adjacency matrix. |  |
| **2.5 Trees**   * Know that a tree is a connected, undirected graph with no cycles   Note that a tree does not have to have a root.   * Know that a rooted tree is a tree in which one vertex has been designated as the root. A rooted tree has a parent/child relationship between nodes. The root is the only node with no parents and all other nodes are descendants of the root. * Know that a binary tree is a rooted tree in which each node has at most two children. * A common application application of a binary tree is as a binary search tree. * Be familiar with typical uses of rooted trees |  |
| **2.6 Hash Tables**   * Concept of a hash table   A hash table is a data structure that create a mapping between keys and values.   * Uses of a hash table * Be able to apply simple hashing algorithms * Know what is meant by a collision   A collision occurs when two key values computer the same hash.   * Know how collisions are handled using rehashing |  |
| **2.7.1 Dictionaries**   * Be familiar with the concept of a dictionary.   A collection of key-value pairs in which the value is accessed via the associated key.   * Be familiar with simple applications of dictionaries, for example information retrieval, and have experience of using a dictionary data structure in a programming language * For example, the document “The green, green grass grows” would be represented by the dictionary: {‘grass’:1,’green’:2,’grows’:1,’the’:1} ignoring letter case. |  |
| **2.8.1 Vectors**  Be familiar with the concept of a vector and the following notations for specifying a vector:   * A **list** of numbers Eg: [2.3,6.8,-1.3,4.5] (=a 1D array) * A **n-vector over R**, written as Rn. Eg the vector above is a 4-vector over R, written as R4 (where R is the set of real numbers) * **A function interpretation** Eg:   0🡪2.3, 1🡪 6.8,2🡪-1.3, 3🡪9  Where 🡪 means map to   * A **geometric point in space**. Eg a pair of (x,y) coordinates such as (1,2) (a 2-vector) * A **dictionary** is a useful way of representing a vector. For instance, the 4-vector above would be represented as {0:2.3, 1: 6.8,2:-1.3:3🡪9}   Where the domain is the set S={0,1,2,3} and the co-domain is R, the set of real numbers.   * A vector can be represented as an **arrow**. Eg the two vector [2.0,3.0] can be represented by an arrow with its tail at the origin and its head at (2.0,3.0) * How to do vector addition * Realise that adding two vectors causes a translation (KC ie movement) * How to do scalar multiplication * Realise that scalar multiplication causes scaling (KC ie stretching or squeezing) * Convex combination of 2 vectors. How to calculate it * Dot product (= scalar product) of two vectors * Applications of the dot product to find the angle between two vectors. |  |