**Theory Answers**

1. **What is the difference between a linked list and an array, and when would you use one over the other? (1 Mark)**

**An array is a collection of elements of a similar data type, whereas a linked list is an ordered collection of elements of the same type in which each element is connected to the next using pointers. Linked lists would be preferred when lots of insertion and deletion is done, since it does not have to shift each element in the list, it only has to change where pointers are pointing. Arrays would be preferred when you want to use as little memory as possible since memory for pointers does not need to be allocated.**

1. What is a dictionary, stack, queue and when would you use it? (3 Marks)

Dictionaries are data structures like JSON objects that have key / value pairs. In a dictionary you cannot have duplicate keys. Dictionaries are useful when you want to order data by grouping it together. Eg. A person with info about his/her age, name, DOB, banking details. Stack stores objects in order they were added (through Push( )), and when you retrieve an object (through Pop( )) it is removed from the stack in a LIFO manner. Queue quite similar to a Stack except it is FIFO. Stacks are useful when you want to process objects added most recently to stack first, Eg. Video game AI that must respond to recent actions first. Queues are used for the opposite result, what was added last gets processed first

1. A hash table is made up of two parts: an array (the actual table where the data to be searched is stored) and a mapping function, known as a hash function. The hash function is a mapping from the input space to the integer space that defines the indices of the array. In other words, the hash function provides a way for assigning numbers to the input data such that the data can then be stored at the array index corresponding to the assigned number. A hash table can insert, remove and search for data with O(1) complexity on average. A collision occurs when two or more values in the input space maps onto the same integer space. Thus having multiple values for the same key. You are tasked to implement a hash table. How would you

a) hash the value if the input is always a number?

The most common hash function is h(k) = k mod m. Where the index of the element is assigned by finding the remainder of the division between the number and the size of the hash table.

b) hash the value if the input is always a string?

Strings can be converted to integers by summing the ASCII values for each character in the string. Eg. John equals 74 + 111 + 104 + 110 = 399. The hash function can then be performed as if the string was a number.

c) Handle collisions in your table? (5 Marks)

One method for resolving collisions looks into the hash table and tries to find another open slot to hold the item that caused the collision. A simple way to do this is to start at the original hash value position and then move in a sequential manner through the slots until we encounter the first slot that is empty.

1. Choose one of the following data structures: Binary-Tree, AVL Tree, Splay Tree, Skip List. Write a description on your chosen data structure, explaining what it is, how it works, why it is useful, an example of execution and when you would use it. The description can be of any size, but penalties will occur if sections are left out. You do not need to cite sources. (10 Marks)

Binary Tree

A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child. A Binary Tree node contains following parts. Data, pointer to left child, pointer to right child.  A binary tree works as follows: every node follows a particular ordering property. This ordering property is true for all nodes n and it states that all left descendants <= n < all right descendants. What defines a node as less than or greater than another node depends on your data type. As mentioned earlier, its time complexity has an average of O(log n) for insert and find as you need not go through each node in the tree. You can divide and conquer by leveraging subsets of the tree. A binary search tree is therefore useful when it comes to filtering through large sets of data. Since you do not have to iterate through the entire set of data, thereby saving a lot of processing time. Binary trees are useful when it comes to storing data in a hierarchical manner, or for decision trees.