Data Structures and Algorithms Assignment 1

1. Describe Python's built-in data structure?

   Python has four built-in data structures. They are Lists, Dictionary, Tuple and Set.

   List: - A list is python's implementation of a dynamic array. They are a ordered collection of data that are mutable. The values are stored in a comma separated manner within a bracket. They support objects of any type of multiple types.

   Eg: List = [1,2,3,4,5]

   Dictionaries: - Dictionaries are used to store values in a key value pair method. They internally make use concepts such as hashing, chaining, open addressing e.tc. They are python's implementation of a Hash Table. They are created with a comma separated list of key: value pairs within curly brackets. They are not stored in a ordered manner, hence rely on hashing methods for random access.

   Eg: numbers = {'one':1,'two':2,'three':3}

   Tuples: - Tuples are a immutable collection of data. They are similar to a list in other aspects.

   Eg: t= (1,2,3,4)

   Set: - They are an unsorted collection of data. They are similar to a list and a tuple, the difference in set is that they only store unique values.

   Eg: primes = {2,3,5,7}

2. Describe the Python user data structure?

   User data structures are data structures that are not present in python as a inbuilt data structure but using python supported concepts user will be able to build these data structures and they are called as user data structures. Examples of this user defined data structures are Linked List, Stack, Queue, Tree and Graph.

   LinkedList: - Linked List is a linear data structure. They are a collection of nodes chained together. Each node will contain two variables one to store the data and the other one to store the addresses memory of the next Node. Direct access cannot be preformed in Linked List as they are not stored in contiguous memory locations.

   Stack: - Stack is a linear data structure that follows the LIFO principle. That is Last In First Out. They support only two function they is to push data or to pop data. According to LIFO they last pushed data will be the first popped data.

   Queue: - Queue is a linear data structure that follows the principle of FIFO (First In First Out). They function like a Queue, the data first inserted will also the data that is first deleted. And the Data will be entered through the front side, deletion happened at the rear end.

Tree: - Trees are non-linear data structures. Data is stored in a tree as a Node. They are the simplest unit in a Tree. The topmost element is called a root, there will mostly be only one root in a Tree. A Node has a Parent node, the node that is above it and a Child Node, a node that is below its. If the Nodes have no children, then they are called as a Leaf.
This type of data structure is highly effective in searching and organizing data.

Graph: Graphs are non-linear data structures consisting of nodes and edges. The nodes containing the data are connected to other nodes with the help of lines called as edges. The nodes in a graph are called as vertices.

3. Describe the stages involved in writing an algorithm

   Design: - Writing an algorithm begins with a Design first. The problem needs to be identified first and then understand them. It is then important to discuss with the stakeholders about the problem and about your approach to solve them.
   The problem is then broken into stages and what happens at each stage is calculated. This flow of operation can be further demonstrated with flowcharts or pseudo code.

   Analyse: - Once the algorithm is written its needs to further analysed for efficiency of the code. Posteriori Analysis and Priori Analysis are means by which an algorithm can be measured for efficiency. The time and space efficiency of the code is measured against the data structures and the compliers used to writer and compile the algorithm. The Big O notations, time complexities and recurrence relations find its application in this phase of the algorithm writing.

   Implement: - After optimal analysis is completed the next step is to write the code using any programming language. The code can be written in java, where the source code is converted to binary code and then compiled with the java compiler and the resulting file can be run in JRE. The java filed is packaged as JAR file and this file can further be deployed.

   Experiment: - In the Experiment phase the algorithm is tested with different values. This is a trial-and-error process where maximum possible different data are used to test the algorithm It is in this process the algorithm is tested for failure. If failed, then the process is repeated hence the stages involved the writing an algorithm is circular. Once this process completed multiple iterations successfully, then we are good to go.

4. Outline the components of a good algorithm?

   - Input: They have a set of Inputs.
   - Output: They produce desired output.
   - Finiteness: The Algorithm should finish in a finite number of steps.
   - Definiteness: They should be definite, as in deterministically result in a solution.
   - Effectiveness: They should effectively produce a solution.

5. Describe the Tree traversal method?

   Tree Traversal is a method in which each node of tree is visited. Unlike linear data structures where the values can be accessed by one way of traversal this non-linear data structures have three different ways its values can be accessed. They are Inorder Traversal, Preorder Traversal, Postorder Traversal.

Inorder Traversal: - Inorder Traversal they policy followed is **Left,Root,Right**. The left subtree of the root node will be traversed first, then we go to the root node and finally the right subtree of the root node will be traversed. The name inorder suggests that its root node will be in between the left subtree and the right subtree.

Eg:- A BST of with values inserted as 1,2,3,4,5. When traversed according to Inorder Traversal will output 4,2,5,1,3.

Preorder Traversal: - This traversal technique follows the policy of **Root,Left,Right.** Here Root node of the tree will be traversed first, then we move to the left subtree and only finally we move to the right subtree.

Eg:- A BST of with values inserted as 1,2,3,4,5. When traversed according to Inorder Traversal will output 1,2,4,5,3.

Postorder Traversal: - This Traversal Technique follows the police of **Left, Right, Root.** First Left Subtree is traversed then we move to the Right Subtree and only finally will the root node will be traversed.

Eg:- A BST of with values inserted as 1,2,3,4,5. When traversed according to Inorder Traversal will output 4,5,2,3,1.

6. Explain the difference between inorder and postorder tree traversal?

Inorder and postorder are two forms of tree traversals policies that are used in a tree to print or access in its values. The only difference between the two policies are the order in which they traversed, namely a Inorder traversal follows the Root, Left, Right pattern of traversal resulting on printing the root node first, then the left node and finally the right node. Whereases Postorder traversal would traverse in the Left, Right, Root pattern that is the left node would be traversed first followed by the right node and only finally shall the root node be traversed. So basically the Inorder traversal begins with the Root node whereases the Postorder would only print the Root node at last.