**1. Stack:** A [stack](https://www.geeksforgeeks.org/stack-data-structure-introduction-program/)is a linear data structure in which elements can be inserted and deleted only from one side of the list, called the **top**. A stack follows the **LIFO** (Last In First Out) principle, i.e., the element inserted at the last is the first element to come out. The insertion of an element into stack is called **push** operation, and deletion of an element from the stack is called **pop** operation. In stack we always keep track of the last element present in the list with a pointer called **top**.

Queue: A queue is a linear data structure in which elements can be inserted only from one side of the list called rear, and the elements can be deleted only from the other side called the front. The queue data structure follows the FIFO (First In First Out) principle, i.e. the element inserted at first in the list, is the first element to be removed from the list. The insertion of an element in a queue is called an enqueue operation and the deletion of an element is called a dequeue operation. In queue we always maintain two pointers, one pointing to the element which was inserted at the first and still present in the list with the front pointer and the second pointer pointing to the element inserted at the last with the rear pointer.

4. (a) **A binary tree** is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller "subtrees" on either side. A null pointer represents a binary tree with no elements -- the empty tree. The formal recursive definition is: a binary tree is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a binary tree.

**A binary search tree** (BST) or "ordered binary tree" is a type of binary tree where the nodes are arranged in order: for each node, all elements in its left subtree are less to the node (<), and all the elements in its right subtree are greater than the node (>).

(b) Traversing in the Binary Tree. Tree traversal is the process of visiting each node in the tree exactly once. Visiting each node in a graph should be done in a systematic manner. If search result in a visit to all the vertices, it is called a traversal.

Binary tree insertion is used to add a new element in a binary search tree at appropriate location. Insert function is to be designed in such a way that, it must node violate the property of binary search tree at each value. Allocate the memory for tree.

(c) A binary tree is made of nodes, where each node contains a "left" reference, a "right" reference, and a data element. The topmost node in the tree is called the root. Every node (excluding a root) in a tree is connected by a directed edge from exactly one other node. This node is called a parent.

A tree has only one root node. Each node can only have one parent but it can have multiple children. A node without a child is called a leaf.

7.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Function | | Use |
| 1) | [strlen(string\_name)](https://www.javatpoint.com/c-strlen) | returns the length of string name. | | |
| 2) | [strcpy(destination, source)](https://www.javatpoint.com/c-strcpy) | copies the contents of source string to destination string. | | |
| 3) | [strcat(first\_string, second\_string)](https://www.javatpoint.com/c-strcat) | concats or joins first string with second string. The result  of the string is stored in first string. | | |
| 4) | [strcmp(first\_string, second\_string)](https://www.javatpoint.com/c-strcmp) | compares the first string with second string.  If both strings are same, it returns 0. | | |
| 5) | [strrev(string)](https://www.javatpoint.com/c-strrev) | returns reverse string. | | |
| 6) | [strlwr(string)](https://www.javatpoint.com/c-strlwr) | returns string characters in lowercase. | | |
| 7) | [strupr(string)](https://www.javatpoint.com/c-strupr) | returns string characters in uppercase. | | |