INSTITUTE OF INFORMATION TECHNOLOGY

JAHANGIRNAGAR UNIVERSITY

Final Assignment

Course Tittle	**:** Data Structure
Course Code	**:** ICT – 2101
Submission Date	**:** 05/08/2021

Submitted To

Dr. M. Abu Yousuf

Professor

IIT – JU

Submitted By

Md. Shakil Hossain

Class Roll – 2023

Exam Roll – 192340

IIT – JU

Answer to the question no - 1

Big-O notation is a theoretical measure of the execution of an algorithm. It defines the implementation time or the amount of use memory.

If one hundred integer elements are chosen at random and inserted into a sorted linked list. to search for an element in that linked list the complete linked list is needed to be traversed it doesn't matter in which order elements are sorted. So in the worst case all the elements will be required to be traversed. So it can be Said that the time complexity for searching an element in a sorted linked list is O(n).

From the characteristics of binary search tree we know that the minimum height of bst is $\log_2 n$ where n is the number of nodes Present in that binary seacrch tree. We also know generally the time complexity of bianary search tree is $O(h)$ where h is the height of that bst.

If the elements are inserted in a random order in any binary search tree the time complexity will be $O(\log_2 n)$ becuase to Search an element where it is in unsorted order then it has to go deep under the height of tree. But when the tree is in Sorted order then it resembles to the Sorted linked list. So the time complexity for Bst when elements are in sorted order is $O(n)$.

Answer to the question no – 2

Strack data structure can be used to Check for balanced Parentheses for the situation mentioned in the question Stacks can check equal Pair or balanced Pair of Parentheses efficiently. Whenever we get an opening Parenthesis we can Push it on the stack and when we get the Corresponding closing Parenthesis we can Pop it. After Performing all Push and Pop operations if at the end of the expression the stack becomes empty then the expression has balanced Parenthesis. From this explanation it can be seen that Stack can be used to solve the Problem given in the question.

Algorithm:

1. ~~Take~~ Initialize on empty stack S.

2. Take a string ~~and~~ with brackets and other symbols like Letters.

3. Traverse the String and read characters until the end of the string.

4. If it is a opening symbol of any bracket Push it onto the Stack.

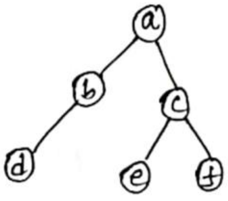
5. If it is a closing symbol ~~of any bracket~~ ~~Push it onto t~~ and the Stack is empty. Print the Position of that symbol. Otherwise Pop the Stack.

6. If the Symbol Popped is not the corresponding opening symbol. then Print the Position of that bracket.

7. At the end of the string if the stack is not empty Print " Unbalanced ".

Answer to the question no - 3

a. We have to design an algorithm for Preorder traverse of the given tree.



It is mentioned that tree node has three fields name 'data' which type is String 'Left_child' and 'right_child'. This left-child and right-child conditain the address of another node. So this two fields data type will consider as node.

```
Struct node
{
    String data;
    Struct node * left_child;
    Struct node * right_child;
};
```

<u>Algorithm:</u>

```
1. Pre-Order (root)
2. If (root = 0)
       Print '.';
       return;

3. Print root → data;
4. Pre-Order (root → left_child);
5. Pre-order (root → right_child);
```

b. We have to create a binary tree from that Pre order format. abd...ce..f.. It is a Preorder format. Here a,b,d,c,e,f Character are the node values and '.' characters are denoting no node there. In Pre-order root of the node comes first then left_child and the right-child

Algorithm:

```
1. Take a string s.
2. Set index i=0
3. Create ()
4. Take a char ch.
       Ch = s[i]
       i++; [i=0 to s.length()]
       if ch='.'
       return 0;
5. Create a newnode
     newnode → data = s[i]
     newnode → left = Create ();
     newnode → right = Create ();
     return newnode;
```

THE END