1- . . . . . is the asymptotic (big-O) time complexity to delete a node from a BST.

2- If the elements "A", "B", "C", "D" and "E" are placed in a queue and are deleted one at a time, ... is the order in which they will be removed.

3- Mathematical description of an object with set of operations on the object is . . . . . .

4- . . . . is a formula to get number of elements in the stack**.**

5- . . . . . is the asymptotic (big-O) time complexity to search in a BST for a specified node.

6- . . . . . is the asymptotic (big-O) time complexity to enqueue an element in a queue.

7- Here is an infix expression: y*m+(h*a^3/b-n)-d. Suppose that we are using the usual stack algorithm to convert the expression from infix to postfix notation, ... is the maximum number of symbols that will appear on the stack at one time during the conversion of this expression.

8- . . . . . is the ADT or data structure that is most appropriate for using in time sharing system in which programs with the same priority waiting to be executed.

9- . . . . . is a high level, language independent, description of a step-by-step process.

10-. . . . . is a specific family of algorithms for implementing an abstract data type.

11-. . . . . is node(s) with the same parent.

12-. . . . . any node, including itself, on the path from the root to the node.

13-. . . . . any node, including itself, on any path from the node to a leaf node.

14-. . . . . number of ancestors of a node excluding itself.

15-. . . . . the length of the longest path from a node to a leaf node.

16-. . . . . is the height of its root node.

17-. . . . . is the number of its children.

18-. . . . . is the number of nodes in a tree.

19- Draw a binary tree from its inorder and preorder traversal sequences given as follow, explain how you draw it step by step:

Inorder : a k b g m h x u n f

Preorder : x b a k m g h u f n

20- Construct a Binary Search Tree by inserting the following sequence of numbers... 10,12,5,4,20,8,7,15 and 13

21- Write a C++ function to implement queue full function.

22- Write a C++ function to implement stack full function.

23- Write a C++ function to implement circular queue full function.

24- Write a C++ function to implement enqueue function.

25- Write a C++ function to implement dequeue function.

26- Write a C++ function to implement push function.

27- Write a C++ function to implement pop function.

28- Write a C++ function to insert a node in to a binary search tree.

29- Write a C++ function to delete a node from a binary search tree.

30- Write a C++ function to traverse a binary search tree in in-order.

31- Write a C++ function to traverse a binary search tree in pre-order.

32- Write a C++ function to traverse a binary search tree in post-order.

33- Write a C++ function to add a new node after a specified node in a single linked list.

34- Write a C++ function to delete a new node after a specified node in a single linked list.

35- Write a C++ function to delete a new node before a specified node in a single linked list.

36- Write a C++ function to add a new node after a specified node.

37- Consider the following pseudocode that uses a stack.

```
declare a stack of characters, and a word to be an input
While (there are more characters in the word to read)
{
     read a character
     push the character on the stack
}
while (the stack is not empty)
{
     pop a character off the stack
     write the character to the screen
}
```

..... is output for input word "Advice"?

38- Consider the following code:

```
struct  Engineering

{   int priority;     char names;    };

struct Software

{   short room;     Engineering info];      };

int main()

{

    Software obj[100];

    ....

    return 0;

}
```

Find SMF for obj[3], where base address=400. Clarify your answer step by step

39- Write the full conditions code of circular queue:

40- Determine step by step the overall asymptotic complexity (Big-O) of the following code statement by statement.

```
for(i=0;i<n;i++)
{
cout<<"Hello";
        for(j=1;j<n;j=j*3)
cout<<"2nd class";
}
```

Good Luck