Trees – II

Non-Recursive PreOrder Traversal
Height of a binary tree
Binary Search Tree

- **Searching**
- **Insertion**
- **Traversal**
- **Creation**

Preorder tree traversal (Non Recursive version)

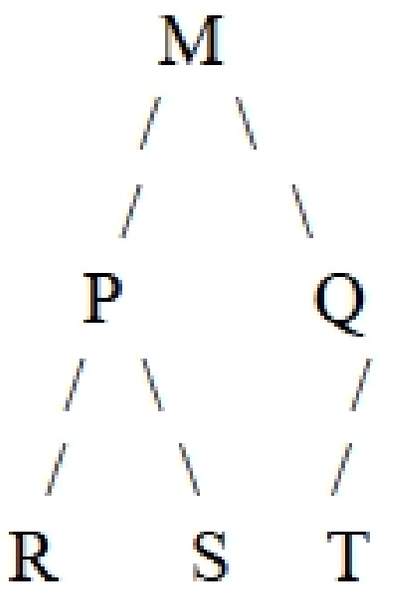
We have earlier seen a recursive algorithm for preorder traversal. In a preorder traversal, we visit the root node first, then we visit its left subtree (all the nodes) and finally visit its right subtree. In this section we shall develop a non-recursive algorithm for the preorder traversal.
Since we can visit only one node at a time, we shall make use of a *stack* to store the other nodes to be visited later.

To start with we push the root node on the stack.
Then we push the right child and the left child of the current node on a stack recursively.
Then we pop them and print them. Look at the following algorithm:

```
*p = root;
if (p != NULL)
{
   push(p);
   while ( stack not empty)     {
        p = pop stack;
           printf("%d ",p->data);
        if ( p->right_child != NULL)
            push ( p-> right_child);
        if (p ->left_child != NULL)
           push (p -> left_child);
       }
}
```

/* note the left child is pushed on top of right node, so that it gets popped up first */

Consider the following tree



Non Recursive Preorder Traversal:

Stack position	Popped values
M	_ _ _
_ _	M
Q P	_ _ _
Q	M P
Q S R	_ _ _
Q S	M P R
Q	M P R
-	M P R S Q
T	_ _ _
-	M P R S Q T

Height of a binary tree:

The height of a tree is given by the height of the root node, which is a count of number of nodes on the longest path from the root to the leaf node. In the following tree the longest