3) Write a program.
a) To construct Binary Search tree.
b) Traverse the tree using inorder, postorder
→ preorder.
c) Display the element in the tree.

```
#include <stdio.h>
#include <stdlib.h>

struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
};

struct TreeNode * createNode (int val){
    struct TreeNode * newnode = (struct TreeNode*)
    malloc (sizeof (struct TreeNode));
    newNode -> val = val;
    newnode -> left = NULL;
    newnode -> right = NULL;
    return newnode;
}

struct TreeNode* insert (struct TreeNode * root,
    int val) {
if (root = NULL) {
        return createNode (val);
}
if (val < root -> val) {
        root -> left = insert (root -> left, val);
}
```

```
else if (val > root -> val){
    root -> right = insert(root -> right, val);
}
return root;
}

void inorderTraversal (struct TreeNode* root){
    if (root == NULL){
        return;
    }
    inorder Traversal (root -> left);
    printf ("%d", root -> val);
    inorder Traversal (root -> right);
}

void preorder Traversal (struct TreeNode* root){
    if (root == NULL){
        return;
    }
    printf ("%d", root -> val);
    preorder Traversal (root -> left);
    preorder Traversal (root -> right);
}

void postorder Traversal (struct TreeNode* root){
    if (root == NULL){
        return;
    }
    postorder Traversal (root -> left)
    postorder Traversal (root -> right)
    printf ("%d", root -> val);
}
```

```
void display Tree (BinaryTreeNode * root){
    printf ("Elements in the tree");
    inorder Traversal (root);
    printf ("\n")
}
```

OUTPUT :

1) Insert
2) Preorder
3) Inorder
4) postorder
5) display.

Insert → 7, 5, 1, 2, 3, 6, 0, 9, 4, 2

Display :(

Preorder → 7 5 1 0 3 2 4 6 8 9

post → 0 2 4 3 1 6 5 9 8 7

Inorder → 0 1 2 3 4 5 6 7 8 9

1.

5

19.02.24