* LOB-10 : * Binary Search Tree :

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

struct node {
        int info;
        struct node *rlink;
        struct node *llink;
        };

typedef struct node *NODE;

NODE getnode() {
        NODE x;
        x = (NODE) malloc (sizeof (struct node));

if (x==NULL) {
                printf("memory full");
                exit(0);
                }

        return x;
}

void freenode (NODE x)
{
        free(x);
}
```

```
NODE insert(NODE root, int item) {

    NODE temp, cur, prev;
    char direction[10];
    int i;
    temp = getnode();
    temp->info = item;
    temp->llink = NULL;
    temp->rlink = NULL;

    if (root == NULL)
        return temp;

    pf("\n give direction : ");
    sf("%s", direction);
    prev = NULL;
    cur = root;

    for (i=0; i<strlen(direction) && cur != NULL; i++)
    {
        prev = cur;
        if (direction[i] == 'l') {
            cur = cur->llink;
        else
            cur = cur->rlink;
        }
    if (cur != NULL || i != strlen(direction))
    {
        printf("insertion not possible");
        freenode(temp);
        return (root);
```

```
if (cur == NULL) {
        if (direction[i-1] == 'l')
            prev->llink = temp;
        else
            prev->rlink = temp;
    }
    return (root);
}

void preorder(NODE root) {
    if (root != NULL) {
        pf("The item is: %d\n", root->info);
        preorder(root->llink);
        preorder(root->rlink);
    }
}

void inorder(NODE root) {
    if (root != NULL) {
        inorder(root->llink);
        pf("The item is: %d\n", root->info);
        inorder(root->rlink);
    }
}
```

```
void postorder(NODE root){
    if(root!=NULL){
        postorder(root->llink);
        postorder(root->rlink);
        printf("\nThe item is :%d",
                root->info);
    }
}

void display(NODE root,int i){
    int j;
    printf("\n\n");
    if(root!=NULL){
        display(root->rlink,i+1);
        for(j=1;j<=i;j++)
            pf("  ");
        printf("%d\n",root->info);
        display(root->llink,i+1);
    }
}
```

```
int main ()
{
    NODE root = NULL;
    int choice, i, items;

for (;;)
{
    pf("1.insert 2.preorder 3.inorder
        4.postorder 5 display");
pf("enter choice");
sf("%d", &choice);

switch (choice)
{
case 1: pf("enter item");
        sf("%d", &items);
        root = insert(item, root);
        break;

case 2: if (root == NULL)
        {
            pf("tree is empty");
        }
        else
        {
            pf("given tree");
            display(root, 0);
            pf("\n preorder traversal is \n");
            preorder(root);
        }
        break;
```

```
case 3: if (root == NULL)
        {
            pf("tree is empty");
        }
        else {
            pf("given tree is");
            display(root, 1);
            pf("\n inorder traversal is \n");
            inorder(root);
        }
        break;

case 4: if (root == NULL)
        {
            pf("tree is empty");
        }
        else {
            pf("given tree");
            display(root, 1);
            pf("\n postorder traversal is");
            postorder(root);
        }
        break;

case : display(root, 1);
       break;

default : exit(0);

    }
}
```