Practical no -9

Aim: To write a program to Create a binary search tree and do the following traversals :

(i)In-order (ii)Pre-order (iii)Post-order

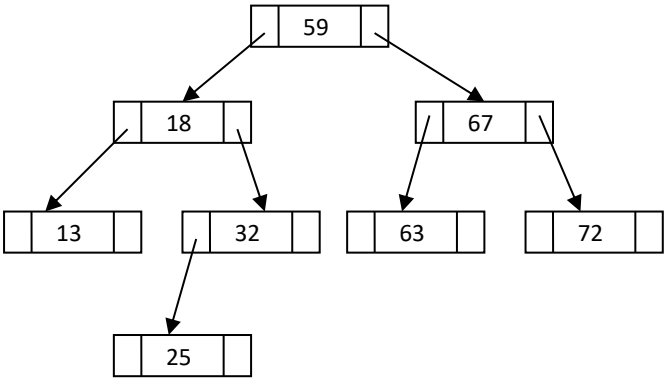
Binary search tree:-

Binary search tree is a binary tree that is either empty or in which each node contains a data value that satisfies the following:

a) All data values in the left subtree are smaller than the data value in the root.
b) The data value in the root is smaller than all values in its right subtree.
c) The left and right subtrees are also binary search tees.

This structure allows us to quickly search for a particular value.

Example:



Binary Tree Traversal Techniques

Traversing of tree means visiting each and every nodes of the binary tree exactly ones called as traversing.

there are three ways to traverse the binary tree,

- Inorder traversing
- Preorder Traversing
- Postorder traversing

Traversing Algorithm and functions:

Inorder Traversing Algorithm:

Steps:1. Traverse the left subtree in inorder manner

2. Process the root

3. Traverse the Reft subtree in inorder manner

Preorder Traversing: Algorithm:

Steps:1. Process the root

2. Traverse the left subtree in preorder manner

3. Traverse the Reft subtree in preorder manner

Postorder Traversing Algorithm:

Steps:

1. Traverse the left subtree in postorder manner
2. Traverse the Reft subtree in postorder manner
3. Process the root

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct bstnode
{
        int data;
        struct bstnode *l_link;
        struct bstnode *r_link;
}*root=NULL,*temp=NULL;


void search(struct bstnode *t);
void inorder(struct bstnode *t);
```

```
void preorder(struct bstnode *t);
void postorder(struct bstnode *t);
void main()
{
        int node_data,choice,yes_no;
        clrscr();
        printf("1.Insert the node \n 2.Inorder Traversal \n 3.Preorder Traversal \n 4. Postorder
        Traversal \n 5.Exit \n");
        do{
        printf("Enter your choice-");
        scanf("%d",&choice);
        switch(choice)
          {
                case 1:
                        printf("Enter the data you want to insert in Node");
                        scanf("%d",&node_data);
                        temp=(struct bstnode*)malloc(1*sizeof(struct bstnode));
                        temp->data=node_data;
                        temp->l_link=temp->r_link=NULL;
                        if(root==NULL)
                        root=temp;
                        else
                        search(root);
                        break;
                case 2:
                        printf("\n Inorder sequence of BST-\n");
                        inorder(root);
                        break;
                case 3:
                        printf("\n Preorder sequence of BST-\n");
```

```
                preorder(root);
                break;
        case 4:
                printf("\n Postorder sequence of BST-\n");
                postorder(root);
                break;
        default:
                printf("You have entered wrong choice\n");
    }
printf("\n do you want to continue press 1 if yes");
scanf("%d",&yes_no);
}while(yes_no==1);
getch();
}
void search(struct bstnode *t)
{
    if((temp->data>t->data)&&(t->r_link!=NULL))
        search(t->r_link);
    else
        if((temp->data>t->data)&&(t->r_link==NULL))
            t->r_link=temp;
        else
            if((temp->data<t->data)&&(t->l_link!=NULL))
                search(t->l_link);
            else
                if((temp->data<t->data)&&(t->l_link==NULL))
                    t->l_link=temp;
}
void inorder(struct bstnode *t)
```

```
{
        if(root==NULL)
                {
                 printf("\n No element in a tree");
                 return;
                }
        if(t->l_link!=NULL)
                 inorder(t->l_link);
        printf("%d->",t->data);
        if(t->r_link!=NULL)
                  inorder(t->r_link);
}
void preorder(struct bstnode *t)
{
        if(root==NULL)
        {
         printf("\n No element in a tree");
         return;
        }
        printf("%d->",t->data);
        if(t->l_link!=NULL)
                 preorder(t->l_link);
        if(t->r_link!=NULL)
                 preorder(t->r_link);
}
void postorder(struct bstnode *t)
{
        if(root==NULL)
        {    printf("\n No element in a tree");
             return;
```

```
        }
        if(t->l_link!=NULL)
                postorder(t->l_link);
        if(t->r_link!=NULL)
                postorder(t->r_link);

        printf("%d->",t->data);

}
```

Output: