```
/*"Given a binary tree represented as linked structure, traverse that binary tree using (i) Preorder(or Depth-first) Traversal, (ii) In-order Traversal, (iii) Post-order Traversal using Recursive Algorithm."*/
#include<stdio.h>
#include<conio.h>
struct nodetype
{
        int data;
        struct nodetype *left,*right;
};
typedef struct nodetype NODE;
        NODE *root = NULL;
void insert(int);
void inorder(NODE*);
void preorder(NODE*);
void postorder(NODE*);
void main()
{
        int n,m;
        clrscr();
        do
        {
                printf("\n\t 1. insert");
                printf("\n\t 2. inorder");
                printf("\n\t 3. preorder");
                printf("\n\t 4. postorder");
                printf("\n\t 5. Exit");
                printf("\n\t Enter your choice:-");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:
                                printf("\n\t Enter value to insert:-");
                                scanf("%d",&m);
                                insert(m);
                                break;

                        case 2:
                                inorder(root);
                                break;
```

```
                        case 3:
                                preorder(root);
                                break;
                        case 4:
                                postorder(root);
                                break;
                }
        }while(n!=5);
}// void main() over.
void insert(int n)
{
        NODE *new1,*temp,*prev;
        new1=(NODE*)malloc(sizeof(NODE));
        new1->left=NULL;
        new1->data=n;
        new1->right=NULL;
        if(root==NULL)
        {
                root=new1;
                return;
        }
        temp=prev=root;
        while(temp!=NULL)
        {
                if(temp->data>n)
                {
                        prev=temp;
                        temp=temp->left;
                }
                else
                {
                        prev=temp;
                        temp=temp->right;
                }
        }

        if(prev->data<=n)
        {
                prev->right=new1;
        }
```

```
        else
        {
                prev->left=new1;
        }
        return;
}//insert over
void inorder(NODE *temp)
{
        if(temp!=NULL)
        {
                inorder(temp->left);
                printf("%d",temp->data);
                inorder(temp->right);
        }
        return;
}
void preorder(NODE *temp)
{
        if(temp!=NULL)
        {
                printf("%d",temp->data);
                preorder(temp->left);
                preorder(temp->right);
        }
        return;
}
void postorder(NODE *temp)
{
        if(temp!=NULL)
        {
                postorder(temp->left);
                postorder(temp->right);
                printf("%d",temp->data);
        }
        return;
}
```