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

#define true 1
#define false 0
typedef struct node *Nodeptr;
struct node{
    int data;
    Nodeptr rchild;
    Nodeptr lchild;
};

#include "StackofNodeptr.h"

Nodeptr getnode(){
    return
((Nodeptr)malloc(sizeof(struct node)));
}

Nodeptr CreateBinaryTree(int item){
    int x;

    if (item!=-1)
    {
        Nodeptr temp = getnode();
        temp->data = item;

        printf("Enter the lchild of
%d :",item);
        scanf("%d",&x);
        temp->lchild =
CreateBinaryTree(x);
        printf("Enter the rchild of
%d :",item);

        scanf("%d",&x);

        temp->rchild =
CreateBinaryTree(x);
        return temp;
    }
    return NULL;
}

void Inorder(Nodeptr root){
    if (root){
        Inorder(root->lchild);
        printf("%d\n",root->data);
        Inorder(root->rchild);
    }
}
void Preorder(Nodeptr root){
   if (root){
        printf("%d\n",root->data);
        Preorder(root->lchild);
        Preorder(root->rchild);
    }
}
void Postorder(Nodeptr root){
    if (root){
        Postorder(root->lchild);
        Postorder(root->rchild);
        printf("%d ",root->data);

    }
}

void iterative_inorder(Nodeptr root)
{
    Nodeptr cur;
    int done = false;

    //STACK *s = (STACK
*)(malloc(sizeof(STACK));
    STACK *s,s1;
    s= &s1;
    s->top = -1;
    if(root==NULL){
        printf("Empty Tree\n");
        return;
    }

    cur=root;
    while(!done)
    {
        while(cur!=NULL)
        {
            Push(s, cur);
            cur=cur->lchild;
        }
        if(!IsEmptyStack(s))
        {
            cur=Pop(s);
            printf("%d ",cur->data);
            cur=cur->rchild;
        }
        else
            done = true;
    }
}

int main(){

    Nodeptr root = NULL;

    int item;

    printf("Creating the tree [enter
-1 for NULL : \n");
    scanf("%d",&item);
    fflush(stdin);
    root=CreateBinaryTree(item);

    printf("\nInorder Traversal : \n");
    Inorder(root);
    printf("\nPreorder Traversal :
\n");
    Preorder(root);
    printf("\nPostorder Traversal :
\n");
    Postorder(root);

    return 0;
```

}