Iterative Tree Traversals

```c
#include <stdio.h>
#include <stdlib.h>
#include "Queue.h"
#include "Stack.h"

struct Node *root=NULL;

void Treecreate()
{
    struct Node *p,*t;
    int x;
    struct Queue q;
    create(&q,100);

    printf("Eneter root value ");
    scanf("%d",&x);
    root=(struct Node *)malloc(sizeof(struct Node));
    root->data=x;
    root->lchild=root->rchild=NULL;
    enqueue(&q,root);

    while(!isEmpty(q))
    {
        p=dequeue(&q);
        printf("eneter left child of %d ",p->data);
        scanf("%d",&x);
        if(x!=-1)
        {
            t=(struct Node *)malloc(sizeof(struct Node));
            t->data=x;
            t->lchild=t->rchild=NULL;
            p->lchild=t;
            enqueue(&q,t);
        }
        printf("eneter right child of %d ",p->data);
        scanf("%d",&x);
        if(x!=-1)
        {
            t=(struct Node *)malloc(sizeof(struct Node));
            t->data=x;
            t->lchild=t->rchild=NULL;
            p->rchild=t;
            enqueue(&q,t);
        }
    }
```

```
}


void IPreorder(struct Node *p)
{
    struct Stack stk;
    Stackcreate(&stk,100);

    while(p || !isEmptyStack(stk))
    {
        if(p)
        {
            printf("%d ",p->data);
            push(&stk,p);
            p=p->lchild;
        }
        else
        {
            p=pop(&stk);
            p=p->rchild;
        }
    }
}

void IInorder(struct Node *p)
{
    struct Stack stk;
    Stackcreate(&stk,100);

    while(p || !isEmptyStack(stk))
    {
        if(p)
        {
            push(&stk,p);
            p=p->lchild;
        }
        else
        {
            p=pop(&stk);
            printf("%d ",p->data);
            p=p->rchild;
        }
    }
}



int main()
```

```
{
    Treecreate();

    IPreOrder(root);
    IInorder(root);

    return 0;
}
```