Binary Tree Create

```
#include <stdio.h>
#include <stdlib.h>
#include "Queue.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 Preorder(struct Node *p)
{
    if(p)
    {
        printf("%d ",p->data);
        Preorder(p->lchild);
        Preorder(p->rchild);
    }
}

void Inorder(struct Node *p)
{
    if(p)
    {
        Inorder(p->lchild);
        printf("%d ",p->data);
        Inorder(p->rchild);
    }
}
void Postorder(struct Node *p)
{
    if(p)
    {
        Postorder(p->lchild);
        Postorder(p->rchild);
        printf("%d ",p->data);
    }
}
```

```c
int main()
{
    Treecreate();
    Preorder(root);
    printf("\nPost Order ");
    Postorder(root);


    return 0;
}
```

Queue Header File

```c
struct Node
{
    struct Node *lchild;
    int data;
    struct Node *rchild;
};

struct Queue
{
    int size;
    int front;
    int rear;
    struct Node  **Q;
};

void create(struct Queue *q,int size)
{
    q->size=size;
    q->front=q->rear=0;
    q->Q=(struct Node **)malloc(q->size*sizeof(struct
Node *));
}

void enqueue(struct Queue *q,struct Node *x)
{
    if((q->rear+1)%q->size==q->front)
        printf("Queue is Full");
    else
    {
        q->rear=(q->rear+1)%q->size;
        q->Q[q->rear]=x;
    }
}

struct Node * dequeue(struct Queue *q)
{
    struct Node* x=NULL;

    if(q->front==q->rear)
        printf("Queue is Empty\n");
    else
    {
        q->front=(q->front+1)%q->size;
        x=q->Q[q->front];
    }
    return x;
}

int isEmpty(struct Queue q)
{
    return q.front==q.rear;
}
```