```
/*the program is to create a binary tree... looks for an ordered tree..
all
left nodes are less than root and right bigger than root.. no
duplication*/

#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <process.h>

struct tree
{
int data;
struct tree *left;
struct tree *right;
};
struct tree *btree=NULL;
struct tree * insert(struct tree *btree, int x);
void inordtr(struct tree*);
void posordtr(struct tree*);
void preordtr(struct tree*);
int degree(struct tree*);

void main(void)
{
int n,i,x;
clrscr();
i=1;
printf ("\nEnter total no. of nodes: ");
scanf("%d",&n);
while(i<=n)
{
        printf(" \nEnter the Node no. %d: ",i);
        scanf("%d",&x);
        btree=insert(btree, x);
        i=i+1;
}
clrscr();
printf("\n\nIn Order\n\n");
inordtr(btree);
printf("\n\nPre Order\n\n");
preordtr(btree);
printf("\n\nPost Order\n\n");
posordtr(btree);

n=degree(btree);
printf("The degree is %d", n);
getch();
}
struct tree* insert(struct tree *btree, int n)
{
if (btree==NULL)
{
        btree=(struct tree*)malloc(sizeof(struct tree));
        btree->data=n;
        btree->left=NULL;
        btree->right=NULL;
}
```

```
else
{
        if(n<btree->data)
                btree->left=insert(btree->left, n);
        else
                if(n>btree->data)
                        btree->right=insert(btree->right,n);
                else
                        if(n==btree->data)
                                {
                                printf("Duplication\n");
                                exit(0);
                                }
}
return(btree);
}


void inordtr(struct tree *b)
{
if (b!=NULL)
{
inordtr(b->left);
printf("%d ->",b->data);
inordtr(b->right);
}
return;
}



void preordtr(struct tree *b)
{
if (b!=NULL)
{
printf("%d ->",b->data);
preordtr(b->left);
preordtr(b->right);
}
return;
}


void posordtr(struct tree *b)
{
if (b!=NULL)
{
posordtr(b->left);
posordtr(b->right);
printf("%d ->",b->data);
}
return;
}


int degree(struct tree *t)
{
        int lt, rt;

        if (t == NULL) return 0;

        lt = 1 + degree(t->left);
        rt = 1 + degree(t->right);

        if (lt > rt)
                return lt;
        else
                return rt;

}
```