Tree Traversal Code

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

struct node {
   char data;
   struct node *lptr;
struct node* rptr;
};

void preorder(struct node*);
void inorder(struct node*);
void postorder(struct node*);

void main()
{
struct node *a,*b,*c,*d,*e,*f,*g;
a = (struct node*)malloc(sizeof(struct node));
b = (struct node*)malloc(sizeof(struct node));
c = (struct node*)malloc(sizeof(struct node));
d=(struct node*)malloc(sizeof(struct node));
e=(struct node*)malloc(sizeof(struct node));
f=(struct node*)malloc(sizeof(struct node));
g=(struct node*)malloc(sizeof(struct node));
```

```
a->data='A';
a->lptr=b;
a->rptr=d;

b->data='B';
b->lptr=c;
b->rptr=NULL;

c->data='C';
c->lptr=NULL;
c->rptr=NULL;

d->data='D';
d->lptr=e;
d->rptr=g;

e->data='E';
e->lptr=NULL;
e->rptr=f;

f->data='F';
f->lptr=NULL;
f->rptr=NULL;

g->data='G';
g->lptr=NULL;
g->rptr=NULL;
```

```
printf("preorder is\n");
preorder(a);
printf("\ninorder is\n");
inorder(a);
printf("\npostorder is\n");
postorder(a);
}

void preorder(struct node*t)
{
if(t==NULL)
{
printf("empty tree");
}
else
printf("%c",t->data);
if(t->lptr!=NULL)
{
preorder(t->lptr);
}
if(t->rptr!=NULL)
{
preorder(t->rptr);
}
}
```

```
void inorder(struct node*t)
{
if(t==NULL)
{
printf("Empty tree");
}
if(t->lptr!=NULL)
{
inorder(t->lptr);
}
printf("%c",t->data);
if(t->rptr!=NULL)
{
inorder(t->rptr);
}
}

void postorder(struct node*t)
{
if(t==NULL)
{
printf("Empty tree");
}
if(t->lptr!=NULL)
{
postorder(t->lptr);
}
```

```
if(t->rptr!=NULL)
{
postorder(t->rptr);
}
printf("%c",t->data);
}
```

Output:

```
preorder is
ABCDEFG
inorder is
CBAEFDG
postorder is
CBFEGDA
```