```
//2020240567  大数据 201  苏恒

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

typedef char DataType;
typedef struct Node {
	DataType data;
	struct Node *lchild, *rchild;
} BTNode ;
typedef BTNode* pBiTree;

void InitialBiTree( pBiTree *pRoot );
void VisitNode( pBiTree root );
void PreOrder( pBiTree root );
void InOrder( pBiTree root );
void PostOrder( pBiTree root );
pBiTree InsertLeftNode( pBiTree pCurNode, DataType x );
pBiTree InserRightNode( pBiTree pCurNode, DataType x );
void CreateBiTree( pBiTree *pRoot );
void DestroyBiTree( pBiTree root );

int main(){
	pBiTree root;
	CreateBiTree( &root );
	printf("\n  前序遍历：");
	PreOrder( root );
	printf("\n");
	printf("  中序遍历：");
	InOrder( root );
	printf("\n");
	printf("  后序遍历：");
	PostOrder( root );
	printf("\n");
	DestroyBiTree( root );
	return 0;
}

void InitialBiTree( pBiTree *pRoot){
	*pRoot = NULL;
	*pRoot = ( pBiTree )malloc( sizeof( BTNode ) );
}
pBiTree InsertLeftNode( pBiTree pCurNode, DataType x ) {
	if( pCurNode == NULL ){
		pCurNode = ( pBiTree )malloc( sizeof( BTNode ) );
		pCurNode->data = x;
		pCurNode->lchild = pCurNode->rchild = NULL;
	}
	else {
		pCurNode->lchild = InsertLeftNode( pCurNode->lchild, x );
	}
	return pCurNode;
	if( !(InsertLeftNode( pCurNode, x)))
		return NULL;
}
pBiTree InsertRightNode( pBiTree pCurNode, DataType x ) {
	if( pCurNode == NULL ){
		pCurNode = ( pBiTree )malloc( sizeof( BTNode ) );
		pCurNode->data = x;
		pCurNode->lchild = pCurNode->rchild = NULL;
```

```
        }
        else {
                pCurNode->rchild = InsertRightNode( pCurNode->rchild, x );
        }
        return pCurNode;
        if( !(InsertRightNode( pCurNode, x)))
                return NULL;
}
void CreateBiTree( pBiTree *pRoot ) {
        InitialBiTree( pRoot );
        if( ( *pRoot ) == NULL ) exit( 0 );
        ( *pRoot )->data = 'A';
        ( *pRoot )->lchild = ( *pRoot )->rchild = NULL;
        ( *pRoot )->lchild = InsertLeftNode( ( *pRoot )->lchild, 'B' );
        ( *pRoot )->lchild->lchild = InsertLeftNode( ( *pRoot )->lchild->lchild, 'E' );
        ( *pRoot )->lchild->lchild->rchild = InsertRightNode( ( *pRoot )->lchild->lchild->rchild, 'F'
 );
        ( *pRoot )->lchild->rchild = InsertRightNode( ( *pRoot )->lchild->rchild, 'C' );
        ( *pRoot )->lchild->rchild->rchild = InsertRightNode( ( *pRoot )->lchild->rchild->rchild,
'D');
        ( *pRoot )->lchild->rchild->rchild->lchild = InsertLeftNode( ( *pRoot )->lchild->rchild
->rchild->lchild,'G');
}
void PreOrder( pBiTree root) {
        if( root != NULL ) {
                VisitNode( root );
                PreOrder( root->lchild );
                PreOrder( root->rchild);
        }
}
void InOrder( pBiTree root ) {
        if( root != NULL ) {
                InOrder( root->lchild );
                VisitNode( root );
                InOrder( root->rchild );
        }
}
void PostOrder( pBiTree root ) {
        if( root != NULL ) {
                PostOrder( root->lchild );
                PostOrder( root->rchild );
                VisitNode( root );
        }
}
void VisitNode( pBiTree root ) {
        printf( "%c ", root->data );
}
void DestroyBiTree( pBiTree root ) {
        if( root == NULL )
                return;
        DestroyBiTree(root->lchild);
        DestroyBiTree(root->rchild);
        free( root );
}
```