#include<stdio.h>

#include<malloc.h>

typedef char datatype;

typedef struct node

{

datatype data;

struct node \*lchild,\*rchild,\*parent;

} bintnode;

typedef bintnode \*bintree;

typedef struct stack

{

bintree data[100];

int tag[100];

int top;

}seqstack;

void push(seqstack \*s,bintree t)

{

s->data[s->top] = t;

s->top++;

}

bintree pop(seqstack \*s)

{

if(s->top != 0)

{

s->top--;

return s->data[s->top];

}

else

return NULL;

}

//按照前序遍历的顺序创建一颗给定的二叉树

bintree createbintree()

{

char ch;

bintree t;

if((ch = getchar())=='#')

t = NULL;

else

{

t = (bintree)malloc(sizeof(bintnode));

t->data = ch;

t->lchild = createbintree();

t->rchild = createbintree();

}

return t;

}

//bintree createbintreeinorder()

//{

// char ch;

// bintree t;

// if((ch = getchar())=='#')

// {

// t = NULL;

// }

// else

// {

// t = (bintree)malloc(sizeof(bintnode));

// t->lchild = createbintreeinorder();

// t->data = ch;

// t->rchild = createbintreeinorder();

// }

// return t;

//}

//二叉树前序遍历非递归实现

void preorder(bintree t)

{

seqstack s;

s.top = 0;

while(t || s.top!=0)

{

if(t)

{

printf("%c ",t->data);

push(&s,t);

t = t->lchild;

}

else

{

t = pop(&s);

t = t->rchild;

}

}

}

//二叉树中序非递归遍历实现

void inorder(bintree t)

{

seqstack s;

s.top = 0;

while(t || s.top != 0)

{

if(t)

{

push(&s,t);

t = t->lchild;

}

else

{

t = pop(&s);

printf("%c ",t->data);

t = t->rchild;

}

}

}

//二叉树后序遍历非递归实现

void postorder(bintree t)

{

seqstack s;

s.top = 0;

while(t || s.top!=0)

{

if(t)

{

s.data[s.top] = t;

s.tag[s.top] = 0;

s.top++;

t = t->lchild;

}

else

{

if(s.tag[s.top - 1] == 1)

{

s.top--;

t = s.data[s.top];

printf("%c ",t->data);

t = NULL;

}

else

{

t = s.data[s.top - 1];

s.tag[s.top - 1] = 1;

t = t->rchild;

}

}

}

}

int main()

{

bintree t = createbintree();

printf("\n二叉树创建完毕，开始遍历!\n\n");

printf("\n\n前序遍历结果:\n");

preorder(t);

printf("\n中序遍历结果:\n");

inorder(t);

printf("\n后序遍历结果:\n");

postorder(t);

return 0;

}