Problem-1

```
#include<iostream>
using namespace std;
typedef enum { Link, Thread }PointerTag;
typedef char TElemType;
typedef struct  Node
{
    TElemType  data;
    Node  *rchild, *lchild;
    PointerTag Ltag = Link, Rtag = Link;
}BiThrNode, *BiThrTree;
BiThrTree pre;

void CreatBiTree(BiThrTree &T)
{
    TElemType c;
    cin >> c;
    if (c == '#')
        T = NULL;
    else
    {
        T = new Node;
        if (!T)
            exit(-1);
        T->data = c;
        CreatBiTree(T->lchild);
        CreatBiTree(T->rchild);
    }
}
/*先序递归线索化二叉树*/
void PreOrderThreading(BiThrTree &T)
{
    if (!T->lchild)
    {
        T->Ltag = Thread;
        T->lchild = pre;
    }
    if (!T->rchild)
        T->Rtag = Thread;
    if (pre&&pre->Rtag == Thread)
```

```
        pre->rchild = T;
    pre = T;
    if (T->Ltag == Link)PreOrderThreading(T->lchild);
    if (T->Rtag == Link)PreOrderThreading(T->rchild);
}
/*去线索化*/
void RemoveThreading(BiThrTree &T)
{

    if (T->Ltag == Thread)
        T->lchild = NULL;
    if (T->Rtag == Thread)
        T->rchild = NULL;
    if (T->Ltag == Link)RemoveThreading(T->lchild);
    if (T->Rtag == Link)RemoveThreading(T->rchild);
}
/*输出树形*/
void ShapeBiThrTree(BiThrTree &T, int Depth)
{
    if (T)
    {
        ShapeBiThrTree(T->rchild, Depth + 1);
        for (int i = 1; i < Depth; i++)
            cout << "     ";
        cout << T->data << T->Ltag << T->Rtag;
        T->Ltag = Link;
        T->Rtag = Link;
        cout << endl;
        ShapeBiThrTree(T->lchild, Depth + 1);
    }
}
/*遍历先序线索二叉树*/
void PreOrderTraverse_Thrt(BiThrTree &T)
{
    BiThrTree  p = T;
    cout << p->data;
    while (p->rchild)
    {
        if (p->Ltag == Link)
            p = p->lchild;
        else
```

```
            p = p->rchild;
        cout << p->data;
    }
}
int main()
{
    BiThrTree T;
    CreatBiTree(T);
    PreOrderThreading(T);
    RemoveThreading(T);
    ShapeBiThrTree(T, 1);
    PreOrderThreading(T);
    PreOrderTraverse_Thrt(T);
    cout << endl;
    return 0;
}
```

Problem-2

```
#include<iostream>
using namespace std;
typedef enum { Link, Thread }PointerTag;
typedef char TElemType;
typedef struct  Node
{
    TElemType  data;
    Node  *rchild, *lchild;
    PointerTag Ltag = Link, Rtag = Link;
}BiThrNode, *BiThrTree;
BiThrTree pre;
void CreatBiTree(BiThrTree &T)
{
    TElemType c;
    cin >> c;
    if (c == '#')
        T = NULL;
    else
    {
        T = new Node;
        if (!T)
            exit(-1);
        T->data = c;
        CreatBiTree(T->lchild);
        CreatBiTree(T->rchild);
    }
}
/*中序递归线索化二叉树*/
void InOrderThreading(BiThrTree &T)
{
    if (T)
    {
        InOrderThreading(T->lchild);
        if (!T->lchild)
        {
            T->Ltag = Thread;
            T->lchild = pre;
        }
        if (!T->rchild)
            T->Rtag = Thread;
        if (pre&&pre->Rtag == Thread)
            pre->rchild = T;
        pre = T;
        InOrderThreading(T->rchild);
    }
}
/*遍历中序线索二叉树*/
void InOrderTraverse_Thrt(BiThrTree &T)
{
    BiThrTree p = T;
    while (p)
    {
        while (p->Ltag == Link)
            p = p->lchild;
        cout << p->data;
        while (p->Rtag == Thread && p->rchild)
        {
            p = p->rchild;
            cout << p->data;
        }
        p = p->rchild;
    }
}
/*寻找目标*/
BiThrNode* SearchNode(BiThrTree &T, TElemType ch, int&flag)
{
    BiThrTree p = T;
    while (p)
    {
        while (p->Ltag == Link)
        {
            p = p->lchild;
        }
        if (p->data == ch)
        {
            flag = 1;
            return p;
        }
        while (p->Rtag == Thread && p->rchild)
        {
            p = p->rchild;
            if (p->data == ch)
            {
                flag = 1;
                return p;
            }
        }
        p = p->rchild;
    }
    flag = 0;
}
/*寻找前驱*/
BiThrNode* InOrderPre(BiThrTree &p)
{
    BiThrTree pre;
    if (p->Ltag == Thread)
        return p->lchild;
    else
    {
        pre = p->lchild;
        while (pre->Rtag == Link)
            pre = pre->rchild;
        return pre;
    }
}
/*寻找后继*/
BiThrNode* InOrderSucc(BiThrTree &p)
{
    BiThrTree succ;
    if (p->Rtag == Thread)
        return p->rchild;
    else
    {
        succ = p->rchild;
        while (succ->Ltag == Link)
            succ = succ->lchild;
        return succ;
    }
}
int main()
{
    BiThrTree T, pre, succ, p;
    int flag = 0;
    TElemType ch;
    CreatBiTree(T);
    cin >> ch;
    InOrderThreading(T);
    InOrderTraverse_Thrt(T);
    cout << endl;
    p = SearchNode(T, ch, flag);
    if (flag == 1)
    {
        succ = InOrderSucc(p);
        cout << "succ is ";
        if (succ == NULL)
            cout << "NULL" << endl;
        else
            cout << succ->data << succ->Rtag << endl;
        pre = InOrderPre(p);
        cout << "prev is ";
        if (pre == NULL)
            cout << "NULL" << endl;
        else
```

```
            cout << pre->data << pre->Ltag << endl;
    }
    else
        cout << "Not found" << endl;
    return 0;
}
```