//1.cpp

#include<iostream>

using namespace std;

class node{

    public:

    int v;

    node \*nxt;

};

class list{

    public:

    node \*head;

    node \*tail;

    list(){

        head=NULL;

        tail=NULL;

    }

    void push(node \*pnn){

        if(tail!=NULL){

            tail->nxt = pnn;

        }

        tail = pnn;

        if(head == NULL){

            head=pnn;

        }

    }

    int calcTot1(int v){

        int i=0,tot=0,j=0;

        node \*trav = this->head;

        while(trav != NULL){

            if(trav->v == v)

                break;

            i++;

            trav = trav->nxt;

        }

        trav = this->head;

        while(trav != NULL){

            if(j >= i-3 && j <= i+3 && j != i)

                tot+=trav->v;

            j++;

            trav = trav->nxt;

        }

        return tot;

    }

    int calcTot2(int v){

        int tot=0;

        node \*tmp[3]={NULL,NULL,NULL},\*trav=head;

        while(trav!=NULL){

            if(trav->v==v){

                tot += tmp[0]->v+tmp[1]->v+tmp[2]->v;

                for(int i=0;i<3 && trav!=NULL;i++){

                    trav=trav->nxt;

                    tot+=trav->v;

                }

                break;

            }else{

                tmp[2]=tmp[1];

                tmp[1]=tmp[0];

                tmp[0]=trav;

            }

            trav=trav->nxt;

        }

        return tot;

    }

    int calcTot3(int v){

        int tot=0;

        node \*tmp,\*trav=head;

        while(trav!=NULL){

            tmp=trav;

            for(int i=0;i<3;i++)

                tmp=tmp->nxt;

            if(tmp->v==v){

                for(int i=0;i<3;i++){

                    tot+=trav->v;

                    trav=trav->nxt;

                }

                trav=trav->nxt;

                for(int i=0;i<3;i++){

                    tot+=trav->v;

                    trav=trav->nxt;

                }

                break;

            }

            trav=trav->nxt;

        }

        return tot;

    }

    ~list(){

        node \*tmp;

        while(head!=NULL){

            tmp = head->nxt;

            delete head;

            head = tmp;

        }

    }

};

int main(){

    list \*l = new list();

    char a='y';

    node \*pnn;

    while(a=='y'){

        pnn = new node;

        cin >> pnn->v;

        pnn->nxt = NULL;

        l->push(pnn);

        cout << "enter another cell ? y/n ";

        cin >> a;

    }

    int v;

    cout << "enter value: ";

    cin >> v;

    cout << l->calcTot1(v) << endl;

    cout << l->calcTot2(v) << endl;

    cout << l->calcTot3(v) << endl;

    return 0;

}

//2.cpp

#include<iostream>

using namespace std;

class node{

    public:

    int v;

    node \*nxt;

};

class list{

    public:

    node \*head;

    node \*tail;

    list(){

        head=NULL;

        tail=NULL;

    }

    void push(node \*pnn){

        if(tail!=NULL){

            tail->nxt = pnn;

        }

        tail = pnn;

        if(head == NULL){

            head=pnn;

        }

    }

    ~list(){

        node \*tmp;

        while(head!=NULL){

            tmp = head->nxt;

            delete head;

            head = tmp;

        }

    }

};

void find(list \*l1,list \*l2){

    node \*t1 = l1->head;

    node \*t2 = l2->head;

    int i=0;

    while (t1 != NULL && t2 != NULL){

        if(t1->v == t2->v)

            cout << i << " , " << t1->v << endl;

        i++;

        t1 = t1->nxt;

        t2 = t2->nxt;

    }

}

int main(){

    list \*l1 = new list();

    char a='y';

    node \*pnn;

    cout << "list 1" << endl;

    while(a=='y'){

        pnn = new node;

        cin >> pnn->v;

        pnn->nxt = NULL;

        l1->push(pnn);

        cout << "another ? y/n ";

        cin >> a;

    }

    list \*l2 = new list();

    a = 'y';

    cout << "list 2" << endl;

    while(a=='y'){

        pnn = new node;

        cin >> pnn->v;

        pnn->nxt = NULL;

        l2->push(pnn);

        cout << "another? y/n ";

        cin >> a;

    }

    find(l1,l2);

    return 0;

}