//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;

    }

    ~list(){

        node \*tmp;

        while(head != NULL){

            tmp = head->nxt;

            delete head;

            head=tmp;

        }

    }

    void attach(node \*pnn){

        if(tail!=NULL)

            tail->nxt=pnn;

        tail=pnn;

        if(head==NULL)

            head=pnn;

    }

    void disp(){

        node \*trav = head;

        while(trav!=NULL){

            cout << trav->v << " ";

            trav=trav->nxt;

        }

        cout << endl;

    }

};

void SplitList\_1(list \*l,int v,list \*l1,list \*l2){

    if(l->head->v==v){

        l1->head=NULL;

        l1->tail=NULL;

        node \*trav=l->head,\*pnn;

        while(trav!=NULL){

            pnn = new node;

            pnn->v = trav->v;

            pnn->nxt=NULL;

            l2->attach(pnn);

            trav=trav->nxt;

        }

        return;

    }

    node \*trav = l->head,\*pnn;

    while(trav->nxt->v != v){

        pnn = new node;

        pnn->v = trav->v;

        pnn->nxt=NULL;

        l1->attach(pnn);

        trav=trav->nxt;

    }

    pnn = new node;//error

    pnn->v = trav->v;

    pnn->nxt=NULL;

    l1->attach(pnn);

    trav=trav->nxt;

    while(trav!=NULL){

        pnn = new node;

        pnn->v = trav->v;

        pnn->nxt=NULL;

        l2->attach(pnn);

        trav=trav->nxt;

    }

}

void SplitList\_2(list \*l,int v,list \*l1,list \*l2){

    if(l->head->v==v){

        l1->head=NULL;

        l1->tail=NULL;

        l2->head = l->head;

        l2->tail = l->tail;

        l->head=NULL;

        l->tail=NULL;

        return;

    }

    node \*trav = l->head;

    while(trav->nxt->v != v){

        trav=trav->nxt;

    }

    l1->tail = trav;

    l1->head = l->head;

    l2->head = trav->nxt;

    l2->tail = l->tail;

    trav->nxt = NULL;

    l->head=NULL;

    l->tail=NULL;

}

int main(){

    int n,v;

    list \*l = new list,\*l1 = new list,\*l2 = new list,\*l3 = new list,\*l4 = new list;

    node \*pnn;

    cin >> n;

    for(int i=0;i<n;i++){

        pnn = new node;

        cin >> pnn->v;

        pnn->nxt = NULL;

        l->attach(pnn);

    }

    cout << "value: ";

    cin >> v;

    SplitList\_1(l,v,l1,l2);

    cout << "Main List: ";

    l->disp();

    cout << "List1: ";

    l1->disp();

    cout << "List2: ";

    l2->disp();

    SplitList\_2(l,v,l3,l4);

    cout << "Main List: ";

    l->disp();

    cout << "List1: ";

    l3->disp();

    cout << "List2: ";

    l4->disp();

    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;

    }

    ~list(){

        node \*tmp;

        while(head != NULL){

            tmp = head->nxt;

            delete head;

            head=tmp;

        }

    }

    void attach(node \*pnn){

        if(tail!=NULL)

            tail->nxt=pnn;

        tail=pnn;

        if(head==NULL)

            head=pnn;

    }

    void disp(){

        node \*trav = head;

        while(trav!=NULL){

            cout << trav->v << " ";

            trav=trav->nxt;

        }

        cout << endl;

    }

};

void ReverseList(list \*l){

    node \*nxt,\*prev=NULL,\*trav=l->head;

    while(trav != NULL){

        if(trav==l->head)

            l->tail=trav;

        nxt = trav->nxt;

        trav->nxt = prev;

        prev = trav;

        trav = nxt;

    }

    l->head = prev;

}

void ReverseList(list \*l,list \*c){

    //error if used attach

    node \*nxt,\*prev=NULL,\*trav=l->head,\*pnn;

    while(trav != NULL){

        pnn = new node;

        pnn->v = trav->v;

        pnn->nxt = prev;

        if(prev==NULL){

            c->tail = pnn;

        }

        prev=pnn;

        trav=trav->nxt;

    }

    c->head = pnn;

}

void ReverseList\_easy(list \*l,list \*c){

    node \*trav=l->head,\*pnn;

    while(trav!=NULL){

        pnn = new node;

        pnn->v = trav->v;

        pnn->nxt=NULL;

        c->attach(pnn);

        trav=trav->nxt;

    }

    ReverseList(c);

}

int main(){

    list \*l=new list,\*l1=new list,\*l2=new list;

    node \*pnn;

    int n;

    cin >> n;

    for (int i=0;i<n;i++){

        pnn=new node;

        cin >> pnn->v;

        pnn->nxt=NULL;

        l->attach(pnn);

    }

    cout << "Main List: ";

    l->disp();

    ReverseList\_easy(l,l1);

    cout << "List1: ";

    l1->disp();

    ReverseList(l,l2);

    cout << "List2: ";

    l2->disp();

    ReverseList(l);

    cout << "Main List: ";

    l->disp();

    return 0;

}

//3.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;

    }

    ~list(){

        node \*tmp;

        while(head != NULL){

            tmp = head->nxt;

            delete head;

            head=tmp;

        }

    }

    void attach(node \*pnn){

        if(tail!=NULL)

            tail->nxt=pnn;

        tail=pnn;

        if(head==NULL)

            head=pnn;

    }

    void disp(){

        node \*trav = head;

        while(trav!=NULL){

            cout << trav->v << " ";

            trav=trav->nxt;

        }

        cout << endl;

    }

    void rotate(int index){

        int i=0;

        node \*trav = head;

        while(trav!=NULL){

            if(i==index-1)

                break;

            trav=trav->nxt;

            i++;

        }

        cout << "trav " << trav->v << endl;;

        tail->nxt = head;

        head = trav->nxt;

        trav->nxt = NULL;

        tail=trav;

    }

};

int main(){

    list \*l=new list;

    node \*pnn;

    int n,index;

    cin >> n;

    for (int i=0;i<n;i++){

        pnn=new node;

        cin >> pnn->v;

        pnn->nxt=NULL;

        l->attach(pnn);

    }

    cout << "List: ";

    l->disp();

    cout << "node: ";

    cin >> index;

    l->rotate(index);

    cout << "List: ";

    l->disp();

    return 0;

}

//4.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;

    }

    ~list(){

        node \*tmp;

        while(head != NULL){

            tmp = head->nxt;

            delete head;

            head=tmp;

        }

    }

    void attach(node \*pnn){

        if(tail!=NULL)

            tail->nxt=pnn;

        tail=pnn;

        if(head==NULL)

            head=pnn;

    }

    void disp(){

        node \*trav = head;

        while(trav!=NULL){

            cout << trav->v << " ";

            trav=trav->nxt;

        }

        cout << endl;

    }

};

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

    int max=l1->head->v,min=l1->head->v,imax,imin,i=0;

    node \*trav=l1->head,\*ma,\*mi,\*pnn;

    while(trav != NULL){

        if(trav->v > max){

            imax=i;

            max=trav->v;

            ma=trav;

        }

        if(trav->v < min){

            imin=i;

            min=trav->v;

            mi=trav;

        }

        trav=trav->nxt;

        i++;

    }

    pnn = new node;

    pnn->v = mi->v;

    pnn->nxt=NULL;

    l2->attach(pnn);

    pnn = new node;

    pnn->v = ma->v;

    pnn->nxt=NULL;

    l2->attach(pnn);

    if(imax<imin){

        trav=ma->nxt;

        while(trav!=mi){

            pnn = new node;

            pnn->v=trav->v;

            pnn->nxt=NULL;

            l2->attach(pnn);

            trav=trav->nxt;

        }

    }else{

        trav=mi->nxt;

        while(trav!=ma){

            pnn = new node;

            pnn->v=trav->v;

            pnn->nxt=NULL;

            l2->attach(pnn);

            trav=trav->nxt;

        }

    }

}

int main(){

    list \*l1=new list,\*l2=new list;

    node \*pnn;

    int n,index;

    cin >> n;

    for (int i=0;i<n;i++){

        pnn=new node;

        cin >> pnn->v;

        pnn->nxt=NULL;

        l1->attach(pnn);

    }

    cout << "list 1: ";

    l1->disp();

    create(l1,l2);

    cout << "list 2: ";

    l2->disp();

    return 0;

}