**INTERSECTION POINT TWO LINKED LISTS**

There are two linked lists. Due to some programming error, the end node of one of the linked list got linked into the second list, forming an inverted Y shaped list. Now it's your job to find the point(Node) at which the two linked lists are intersecting.

**Input Format:**

The first line of input is the size(N1) of the first linked list, followed by its content(Xi). The third line of input is the size(N2) of the second linked list, followed by its content( Yi ).

**Constraints:**

1<=N, N2<=100 0<=Xi, Yi<=100

**Output Format**

A single integer denoting the intersection point of two linked lists. If the linked lists are not intersecting at any point then print -1.

**Sample Input**

5

10 20 30 40 50

4

15 25 40 50

**Sample Output**

40

Program-

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*next;

Node(int d)

{

data=d;

next=NULL;

}

};

void insertAtEnd(Node \*&head,int num)

{

if(head==NULL)

{

head=new Node(num);

return;

}

Node \*tail=head;

while(tail->next!=NULL)

tail=tail->next;

Node \*n=new Node(num);

tail->next=n;

return;

}

int getmidPoint(Node \*head1,Node \*head2,int n,int m)

{

if(n>m)

{

while(n!=m)

{

head1=head1->next;

n--;

}

while(head1!=NULL)

{

Node \*head3=head2;

while(head3!=NULL)

{

if(head1->data==head3->data)

return head1->data;

head3=head3->next;

}

head1=head1->next;

}

}

if(m>n)

{

while(n!=m)

{

head2=head2->next;

m--;

}

while(head1!=NULL)

{

Node \*head3=head2;

while(head3!=NULL)

{

if(head1->data==head3->data)

return head1->data;

head3=head3->next;

}

head1=head1->next;

}

}

if(n==m)

{

while(head1!=NULL)

{

Node \*head3=head2;

while(head3!=NULL)

{

if(head1->data==head3->data)

return head1->data;

head3=head3->next;

}

head1=head1->next;

}

}

return -1;

}

int main()

{

Node \*head1=NULL;

Node \*head2=NULL;

int n,i,num,m;

cin>>n;

for(i=0;i<n;i++)

{

cin>>num;

insertAtEnd(head1,num);

}

cin>>m;

for(i=0;i<m;i++)

{

cin>>num;

insertAtEnd(head2,num);

}

cout<<getmidPoint(head1,head2,n,m);

}