**Section (I): Tracing Problems (Total: 2 marks)**

*In the following tracing question use the following definition for the nodes in the list:*

**class CListNode**

**{**

**public:**

**int info;**

**CListNode \*pNext;**

**};**

**A**

**1**

**2**

**4**

**3**

**Write one statement to do the followings:-**

* Display the info of node (**3**).
* Make node (**3**) points to node (**4**).

**Section (II): Algorithms (Total: 8 marks)**

**Algorithm 1: (8 marks)**

*In the following Algorithm question use the following definition for the nodes in the list:*

**class CNode**

**{**

info

**public:**

pNext

**int info;**

**CNode \*pNext;**

**CNode \*pDown;**

**};**

pDown

Write  **void Display\_And\_Move**() function that do the followings:

* Your function will display the first column in the List, and move it to the last column.

**Note:** Make your function in **O(K)**.

e.g.

T

H1

K

M

**Disp() 🡪 15, 30, 39, 41, 50**

T

H1

**Problem 1: (10 marks)**

Write a main function to do:

* Read a List (L1) from the user.
* Count the number of nodes in (L1)

L1

Head

* Read another List (L2) from the user.
  + Assume that all values of L2 are included in L1, and in the same order of L1.
* Count the number of nodes in (L2)

L2

Head

* Pad (L2) with zeros, to make each node in L2 in the same position like L1.

Head

L2

* In (L1), Check if the unmatched nodes are mirror or not.

40, 75, 12, 12, 75, 40

void main()

{

CList L1,L2;

CNode\* pnn, \* pTrav;

int N, ct1 = 0, ct2 = 0;

cout << "enter n \n";

cin >> N;

for (int j = 0; j < N; j++)

{

pnn = new CNode;

cout << "enter pnn info \n";

cin >> pnn->info;

pnn->pNext = NULL;

L1.Attach(pnn);

ct1++;

}

cout << "enter n \n";

cin >> N;

for (int j = 0; j < N; j++)

{

pnn = new CNode;

cout << "enter pnn info \n";

cin >> pnn->info;

pnn->pNext = NULL;

L2.Attach(pnn);

ct2++;

}

CNode\* pTrav1 = L1.pHead;

CNode\* pTrav2 = L2.pHead;

CNode\* pB= NULL,\*pnn1;

int num = ct1-ct2;

int i = 0,check=0,pos;

CStack S;

while (pTrav2 != NULL)

{

if (pTrav1->info != pTrav2->info)

{

pnn = new CNode;

pnn->info = 0;

pnn->pNext = pB->pNext;

pB->pNext = pnn;

if (i < num)

{

pnn1 = new CNode;

pnn1->info = pTrav1->info;

S.push(pnn1);

}

else

{

if (check == 0) //to know the start of comparison in L1

{

pos = i;

check = 1;

}

}

i++;

}

else

{

pB = pTrav2;

pTrav2 = pTrav2->pNext;

}

pTrav1 = pTrav1->pNext;

}

pTrav1 = L1.pHead;

pTrav2 = L2.pHead;

i = 0;

while (i != pos)

{

pTrav1 = pTrav1->pNext;

pTrav2 = pTrav2->pNext;

i++;

}

int bad = 0;

while (pTrav1 != NULL)

{

if (pTrav2->info == 0)

{

CNode\* pS = new CNode;

pS = S.pop();

if (pTrav1->info != pS->info)

{

bad = 1;

}

}

pTrav1 = pTrav1->pNext;

pTrav2 = pTrav2->pNext;

}

if (bad == 1)

{

cout << "not mirror";

}

else

{

cout << "mirror";

}

}