**Section (I): Tracing Problems (Total: 3 marks)**

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

**class CListNode**

pNext

info

pBack

**{**

**public:**

**int info;**

**CListNode \*pNext;**

**CListNode \*pBack;**

**};**

**A**

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

* Display the info of node (1).

Cout<<A->pBack->pNext->info;

* Display the info of node (2).

Cout<<A->pBack->pNext->pNext->info;

* Make the “pNext” of Node(3) points to the Node (5).

A->pBack->pNext=A;

**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 CListNode**

pFollow

**{**

pBack

**public:**

pNext

**int info;**

**CListNode \*pNext;**

**CListNode \*pBack;**

**CListNode \*pFollow;**

**};**

info

Write a function which called

*void* ***Remove*** *(int Target)*

* Your function will receive a target value to remove it.
* Find the Target node. O(N).
* Remove it & its follow O(1).

**e.g.**

**Remove (17)**

MyList

Head

MyList

Head

void Remove(int target)

{

CNode\* pTrav = pHead, \* pB = NULL, \* pF = NULL, \* pN = NULL, \* ptB = NULL;

while (pTrav != NULL)

{

if (pTrav->info == target)

{

pF = pTrav->pFollow;

pN = pTrav->pNext;

ptB = pF->pBack;

break;

}

pB = pTrav;

pTrav = pTrav->pNext;

}

pB->pNext = pN;

pN->pBack = pB;

pTrav->pNext = pTrav->pBack = NULL;

ptB->pNext = pF->pNext;

pF->pNext->pBack = ptB;

pF->pNext = pF->pBack = NULL;

delete pTrav, pF;

}

**Section (III): Problem Solving (Total: 9 marks)**

**problem 1: (9 marks)**

**Write a main function to do the followings:**

* Read 20 linked lists from the user.

Head

Head

Head

Head

* Ask the user to enter 2 positions **(P1, P2)**.

e.g.

**P1** = 3

**P2** = 1

Head

Head

Head

Head

Then check if the column of (**P2**) is mirror for the column of (**P1**).

**Note:** take from **P2** number of nodes equal to **P1.**

Output:🡪 YES it is mirror (28, 13, 50, **|** 50, 13, 28).

void main()

{

CList L[20];

CNode\* pnn, \* pTrav;

int N, p1, p2, start = 0,end=0;

for (int i = 0; i < 20; i++)

{

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;

L[i].Attach(pnn);

}

}

cout << "enter p1 \n";

cin >> p1;

cout << "enter p2 \n";

cin >> p2;

int ct1 = 0, ct2 = 0;

//to count the no.of nodes for each column

for (int i = 0; i < 20; i++)

{

pTrav = L[i].pHead;

for (int j = 0; pTrav != NULL; j++)

{

if (j == p1)

{

ct1++;

}

if (j == p2)

{

ct2++;

}

pTrav = pTrav->pNext;

}

}

int min = 9999;

if (ct1 < ct2)

{

start = p1;

end = p2;

min = ct1;

}

else

{

start = p2;

end = p1;

min = ct2;

}

//to create stack of the smallest num of nodes

CStack S;

for (int i = 0; i < 20; i++)

{

pTrav = L[i].pHead;

for (int j = 0; j < start && pTrav != NULL; j++)

{

pTrav = pTrav->pNext;

}

pnn = new CNode;

pnn->info = pTrav->info;

pnn->pNext = NULL;

S.push(pnn);

}

int z = 0, bad = 0;

for (int i = 0; i < 20; i++)

{

pTrav = L[i].pHead;

for (int j = 0; j < end && pTrav != NULL; j++)

{

pTrav = pTrav->pNext;

}

CNode\* pS = S.pop();

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

{

bad = 1;

}

z++;

if (z == min)

{

break;

}

}

if (bad == 1)

{

cout << "not mirror";

}

else

{

cout << "mirror";

}

}