**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 following:-**

* Display the info of node (1).

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

* True or False : A 🡪 pBack 🡪 pBack == A

TRUE

* Make the “pNext” of Node(3) points to the Node (4).

A->pBack->pNext->pNext=A;

**Section (II): Algorithms (Total: 7 marks)**

**Algorithm 1: (7 marks)**

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

**class CNode**

pUp

**{**

**public:**

info

pNext

**int info;**

**CNode \*pNext;**

**CNode \*pUp;**

**CNode \*pDown;**

**};**

pDown

Write a function with **O(N)** which called

*CNode\** ***Insert*** *( CNode \* pNN )*

* *The function will insert the pnn in the list.*
* *Create* ***UP*** *node that carries the total of all previous nodes.*
* *Create* ***Down*** *node that carries the total of all coming nodes.*

e.g.

H

Insert (75)

H

**12 + 70 = 82**

**80+85+120+160 = 445**

CNode\* Insert(CNode\* pnn)

{

int tot = 0;

CNode\* pTrav = pHead, \*pB=NULL, \* pU=NULL;

while (pTrav != NULL)

{

tot += pTrav->info;

if (pTrav->info > pnn->info)

{

pB->pNext = pnn;

pnn->pNext = pTrav;

pU = new CNode;

pU->info = tot - (pTrav->info); //total of 1st two nodes without the 80 in eg

pnn->pUp = pU;

pU->pNext = pU->pDown = pU->pUp = NULL;

}

pB = pTrav;

pTrav = pTrav->pNext;

}

//kda kda the info in pnn mesh htet7sb fi el total 3ashan el ptrav htkon 3la el node elly b3deeha

CNode\* pD = new CNode;

pD->info = tot - (pU->info); //total of all nodes without the 1st two nodes in eg

pnn->pDown = pD;

pD->pNext = pD->pDown = pD->pUp = NULL;

return pnn;

}

**Section (III): Problem Solving (Total: 10 marks)**

**Problem 1: (10 marks)**

Write a main function to do:

* + Read 30 Lists from the user.
  + Save the length of each List.

Head

Head

Head

L

Len = 9

Len = 6

Len = 7

Head

Len = 5

Head

Len = 8

* Find the shortest list, cut its 2nd half and past it to new list. **(NL)**
* Find the longest list , cut its 1st half and past it to the same new list. **(NL)**

NL

* Check if the **(NL)** is mirror or not.

void main()

{

CList L[30], newL;

CNode\* pnn, \* pTrav, \* pB = NULL;

int N, max = -9999, min = 9999, ct = 0, posmin = 0, posmax = 0;

cout << "enter n \n";

cin >> N;

for (int i = 0; i < 30; 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);

}

}

for (int i = 0; i < 30; i++)

{

pTrav = L[i].pHead;

ct = 0;

while (pTrav != NULL)

{

ct++;

pTrav = pTrav->pNext;

}

if (ct < min)

{

min = ct;

posmin = i;

}

if (ct > max)

{

max = ct;

posmax = i;

}

}

pTrav = L[posmin].pHead;

int i = 1,ct=0;

while (pTrav!=NULL)

{

if (i > min / 2)

{

ct++;

if (newL.pHead == NULL)

{

newL.pHead = pTrav;

newL.pTail = pTrav;

pB->pNext = pTrav->pNext;

newL.pTail->pNext = NULL;

pTrav = pB;

}

else

{

newL.pTail->pNext = pTrav;

newL.pTail = pTrav;

newL.pTail->pNext = NULL;

pTrav = pB;

}

}

pB = pTrav;

pTrav = pTrav->pNext;

i++;

}

pTrav = L[posmax].pHead;

pB = NULL;

for (int i = 0; i < max / 2 && pTrav != NULL; i++)

{

ct++; //no.of nodes in new L

newL.pTail->pNext = pTrav;

newL.pTail = pTrav;

newL.pTail->pNext = NULL;

pTrav = pB;

pB = pTrav;

pTrav = pTrav->pNext;

}

CStack S;

pTrav = newL.pHead;

for (int i = 0; i < ct / 2 && pTrav != NULL; i++)

{

pnn = new CNode;

pnn->info = pTrav->info;

S.push(pnn);

pTrav = pTrav->pNext;

}

//pTrav in 1st node of second half of newL

int bad = 0;

while (pTrav != NULL)

{

CNode\* pS = S.pop();

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

{

bad = 1;

}

delete pS;

pTrav = pTrav->pNext;

}

if (bad == 1)

{

cout << "not mirror";

}

else

{

cout << "mirror";

}

}