**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->pNext->info;

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

FALSE

* Make the “pBack” of Node(2) points to the Node (1).

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

**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**

**{**

**public:**

info

pNext

**int info;**

**CNode \*pNext;**

**};**

*void* ***Attach*** *( CNode \* pNN )*

* *The function will attach the pNN in the list according to the following protocol:*
  + *In case the pNN carries a positive value*
    - *Then attach it normally at the end*
  + *In case the pNN carries a negative value* 
    - *Insert it after the 1st negative value in the list.*
* Write a function with **O(1)**
* Assume: there is an additional pointer in your CList class called ***pOther***.

e.g.

pTail

pHead

pOther

pHead

pTail

Attach (35)

pOther

Attach (-18)

pOther

pTail

pHead

pHead

pTail

pHead

pTail

Attach (100)

pOther

Attach (65)

pOther

Attach (-10)

pTail

pHead

pTail

pHead

pTail

pHead

pOther

Attach (40)

Attach (25)

pOther

Attach (- 5 )

pOther

void Attach(CNode\* pnn)

{

if (pHead == NULL)

{

pHead = pnn;

pTail = pnn;

if (pnn->info < 0)

{

pOther = pnn;

}

}

else

{

if (pnn->info > 0)

{

pTail->pNext = pnn;

pTail = pnn;

}

else

{

if (pOther == NULL)

{

pTail->pNext = pnn;

pTail = pnn;

pOther = pnn;

}

else

{

pnn->pNext = pOther->pNext;

pOther->pNext = pnn;

}

}

}

}

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

**Problem 1: (10 marks)**

Write a main function to do:

* + Read 30 Lists from the user.

Head

Head

Head

L

Len = 9

Len = 6

Len = 7

Head

Len = 5

Head

Len = 8

* Find the shortest list,
  + Then : from the other lists ,

cut the node at the middle

and past it to the shortest list.

Note: in case the length of the list is even, then the middle node will be the 1st node in the 2nd half.

Head

Head

Head

Head

Head

L

void main()

{

CList L[30];

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

int N, posl = 0,ct=0, min = 9999,half=0;

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

{

cout << "enter N \n";

cin >> N;

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

{

pnn = new CNode;

cout << "enter info pnn \n";

cin >> pnn->info;

pnn->pNext = NULL;

L[j].Attach(pnn);

}

}

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

{

ct = 0;

pTrav = L[j].pHead;

while (pTrav != NULL)

{

ct++;

}

if (ct < min)

{

min = ct;

posl = j;

}

}

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

{

ct = 0;

if (j != posl) //to enter each list except the shortest

{

pTrav = L[j].pHead;

while (pTrav != NULL)

{

ct++;

}

pTrav = L[j].pHead;

for (int i = 0; i < ct/2; i++)

{

pB = pTrav;

pTrav = pTrav->pNext;

}

L[posl].pTail->pNext = pTrav;

L[posl].pTail = pTrav;

pB->pNext = pTrav -> pNext;

pTrav->pNext = NULL;

}

}

}