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

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

**class CListNode**

**{**

**public:**

**int info;**

**CListNode \*pNext;**

**};**

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

* Display the info of node (**1**).

Cout<<A->pNext->info;

**True or False :**

* A 🡪 pNext == A 🡪 pNext 🡪 pNext 🡪 pNext 🡪 pNext.

TRUE

* Make node (**2**) points to node (**3**).

A ->pNext-= A->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:**

**int info;**

**CNode \*pNext;**

**};**

Write  **void Insert\_List\_into\_List** **( CList &L2 )** function that do the following:

* Your function will receive another List (**L2**). (Assume L2 is already sorted)
* Insert the nodes of (*L2*) into (*this*)
* Your function should be in **O(N)**

**Not accepted O(2N).**

e.g.

void Insert\_List\_into\_List(CList& L2)

{

CNode\* pTrav = pHead, \* pB = NULL, \* pTrav2 = L2.pHead;

while (pTrav != NULL && pTrav2 != NULL)

{

if (pTrav->info < pTrav2->info)

{

pB = pTrav;

pTrav = pTrav->pNext;

if (pTrav->pNext = NULL)

{

pTail->pNext = pTrav2;

pTail = pTrav2;

}

}

else

{

if (pB != NULL) //the insert is not at the start of the main list

{

L2.pHead = pTrav2->pNext;

pB->pNext = pTrav2;

pTrav2->pNext = pTrav;

pTrav2 = pTrav2->pNext;

}

else

{

L2.pHead = pTrav2->pNext;

pTrav2->pNext = pTrav;

pHead = pTrav2;

}

}

}

}

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

**Problem 1: (10 marks)**

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

**class CNode**

**{**

**public:**

**int info;**

**CNode \* pNext;**

**CNode \* pUp;**

**};**

Write a main function to do:

* + Read 20 Lists from the user.

According to the following protocol :

* The 1st List will contains only 1 node.
* The 2nd List will increased the length by 2 cells.
* And so on.

e.g.

* Create a newList (NL) such that :
  + For each list get the lowest value. (**Min**)
  + Copy (Min) to (NL) .
  + from the other Lists 🡪 Copy any node > (Min) but upward.

#include <iostream>

using namespace std;

class CNode

{

public:

int info;

CNode\* pNext;

CNode\* pUp;

};

class CList

{

public:

CNode\* pHead;

CNode\* pTail;

CList()

{

pHead = NULL;

pTail = NULL;

}

void Attach(CNode\* pnn)

{

if (pHead == NULL)

{

pHead = pnn;

pTail = pnn;

}

else

{

pTail->pNext = pnn;

pTail = pnn;

}

}

~CList()

{

CNode\* pTrav = pHead;

while (pHead != NULL)

{

pHead = pTrav->pNext;

pTrav->pNext = NULL;

delete pTrav;

pTrav = pHead;

}

}

};

void main()

{

CList L[20], newL;

CNode\* pnn, \*pTrav;

int N=1,min=9999;

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

{

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);

}

N += 2;

}

for (int i = 0; i < 20; i++) //creating the horizontal nodes of new L

{

pTrav = L[i].pHead;

min = 9999;

while (pTrav != NULL)

{

if (pTrav->info > min)

{

min = pTrav->info;

}

pTrav = pTrav->pNext;

}

pnn = new CNode;

pnn->info = min;

pnn->pNext = NULL;

newL.Attach(pnn);

}

CNode\* pTrav2 = newL.pHead;

int pos = 0;

while (pTrav2 != NULL)

{

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

{

if (pos != i)

{

pTrav = L[i].pHead;

while (pTrav != NULL)

{

if (pTrav->info > pTrav2->info)

{

pnn = new CNode;

pnn->info = pTrav->info;

pnn->pNext = NULL;

if (pTrav2->pUp == NULL)

{

pTrav2->pUp = pnn;

pnn->pNext = pnn;

}

else

{

pTrav2->pUp->pNext->pUp = pnn;

pTrav2->pUp->pNext = pnn; //next of 1st up node point to last node

}

}

pTrav = pTrav->pNext;

}

}

}

pTrav2 = pTrav2->pNext;

pos++; //to be able to traverse along ALL other lists

}

}