**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;**

**};**

**A**

**1**

**2**

**4**

**3**

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

* Display the info of node (**3**).

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

* Make node (**1**) points to node (**2**).

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

**True or False :**

* A 🡪 pNext 🡪 pNext 🡪 pNext == A

FALSE

**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:**

pNext

**int info;**

**CNode \*pNext;**

pUp

info

**CNode \*pDown;**

**CNode \*pUp;**

**};**

pDown

Write  **void Remove** **( int value )** function that do the following:

* Your function will receive an integer that represents a value to be found and removed.
* **Note:** Don’t remove negative values.

Assume : The negative values will be only on the row pointed by the (**pHead**).

pHead

pHead

e.g.

**Remove** ( 41 )

pHead

pHead

**Remove** ( 22 )

void Remove(int value)

{

CNode\* pTrav, \* pTu, \* pBu, \* pTd, \* pBd;

pTrav = pHead;

pBu = pBd = NULL;

int check = 0;

while (pTrav != NULL)

{

check = 0;

pTu = pTrav->pUp;

pTd = pTrav->pDown;

while (pTu != NULL)

{

if (pTu->info == value)

{

pBu->pNext = pTu->pNext;

pTu->pNext = NULL;

delete pTu;

check = 1;

break;

}

pBu = pTu;

pTu = pTu->pUp;

}

if (check == 1)

{

break;

}

while (pTd != NULL)

{

if (pTd->info == value)

{

pBd->pNext = pTd->pNext;

pTd->pNext = NULL;

delete pTd;

check = 1;

break;

}

pBd = pTd;

pTd= pTu->pDown;

}

if (check == 1)

{

break;

}

pTrav = pTrav->pNext;

}

}

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

**Problem 1: (10 marks)**

Write a main function to do:

* + Read 30 Lists from the user.
  + Ask the user to select 2 lists.
  + In the 2nd list find 2 nodes equal to

the (1st & last nodes) in the 1st list.

e.g.

List # 0

List # 2

Head

Head

Head

L

Head

Head

* Cut and paste the interval from 2nd list to the 1st list , **but** in reverse manner.

**Note**: make this step in **O(1)**.

Head

Head

Head

Head

Head

void main()

{

CList L[30];

CNode\* pnn;

int N,l1,l2;

cout << "enter N \n";

cin >> N;

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

{

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);

}

}

cout << "enter l1\n";

cin >> l1;

cout << "\n enter l2\n";

cin >> l2;

CNode\* pTrav = L[l2].pHead;

CNode\* p1, \* p2;

while (pTrav != NULL)

{

if (pTrav->info == L[l1].pHead->info)

{

p1 = pTrav;

}

if (pTrav->info == L[l1].pTail->info)

{

p2 = pTrav;

}

}

pTrav = p1->pNext;

while (p1 != p2)

{

//the cut part itself is the O(1), that's how the dr did it

p1->pNext = pTrav->pNext;

pTrav->pNext = L[l1].pTail->pNext;

L[l1].pTail->pNext = pTrav;

pTrav = p1->pNext;

}

}