**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;**

**};**

**B**

**A**

**4**

**3**

**1**

**2**

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

* Display the info of node (**2**).

Cout<<B->pNext->info;

**True or False :**

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

TRUE

* Make node (**1**) points to node (**3**).

B->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**

**{**

K

info

**public:**

**int info;**

pNext

**int K;**

**CNode \*pNext;**

**CNode \*pDown;**

**};**

pDown

Write  **void Attach\_And\_PointBackTo\_kth\_Node** **( CNode \*pnn , int K )** function that do the following:

* Your function will attach the (pnn) at the tail of the List.
* Also, make the (pDown) of the (pnn) points back to kth node that precedes the (pnn).
* Your function should be in **O(N)**

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

e.g.

**Attach** ( 25 , 3 )

pHead

pHead

e.g.

**Attach** ( 70 , 6 )

pHead

pHead

**Note:** if there is no (K) nodes precedes the (pnn) 🡪 simply don’t attach this (pnn).

**Note:** it is up to you to assume there is a (***pTail***) in your list class or not.

void Attach\_And\_PointBackTo\_kth\_Node(CNode\* pnn, int kval)

{

if (pHead == NULL)

{

pHead = pnn;

pTail = pnn;

pnn->K = kval;

}

else

{

pnn->K = kval;

pTail->pNext = pnn;

pTail = pnn;

}

int ct = 1,pos=0;

CNode\* pTrav = pHead;

if (kval != 0)

{

while (pos-ct != kval)

{

if (pTrav == pnn && pos == 0)

{

pos = ct;

ct = 1;

pTrav = pHead; //pTrav start from head again to count till pos

}

else

{

ct++;

pTrav = pTrav->pNext;

}

}

pnn->pDown = pTrav;

}

}

**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 CListNode**

info

**{**

**public:**

pNext

**int info;**

**CListNode \*pNext;**

**CListNode \*pDown;**

**};**

pDown

Write a main function to do:

* Read a Linked List **(L)** form the user, but according to the following protocol :

- the user will determine the number of columns in the list. (ASSUME will be odd)

- in 1st column read 1 down node, AND , last column read also 1 down node.

- in 2ndcolumn read 2 down nodes, AND ,the column before the last read also 2 down nodes.

- and so on.

**e.g.**

L

Tail

Head

* Ask the user to select a column (posC),
  + Display all nodes in this selected column , but in reverse manner.
  + Also display the another column which includes the same length.

e.g.

posC = 2

Tail

Head

L

**Output:**

**17 , 36 , 42 , 50**

**14 , 91 , 37 , 32**

#include <iostream>

using namespace std;

class CNode

{

public:

int info;

CNode\* pNext;

CNode\* pDown;

};

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;

CNode\* pnn, \*pnn1;

int N, posC = 0;

cout << "enter N \n";

cin >> N;

/\* Dr.Ahmed method\*/

int ct=1;

for (int i = 1; i <= N; i++)

{

pnn = new CNode;

cout << "enter info pnn \n";

cin >> pnn->info;

pnn->pNext = NULL;

pnn->pDown = NULL;

L.Attach(pnn);

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

{

pnn1 = new CNode;

cout << "enter info pnn1 \n";

cin >> pnn1->info;

pnn1->pNext = NULL;

pnn1->pDown = NULL;

if (i < (N + 1) / 2)

{

ct++;

}

else

{

ct--;

}

}

}

cout << "enter posC \n";

cin >> posC;

CNode\* pTrav = L. pHead;

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

{

pTrav = pTrav->pNext;

}

/\*when dr solved Q in class, he used an additional data struct\*/

CStack S;

CNode\* pS = pTrav;

while (pS != NULL)

{

pnn = new CNode;

pnn->info = pS->info;

pnn->pNext = pS->pNext = NULL;

S.push(pnn);

pS = pS->pDown;

}

CNode\* pD;

while (S.pHead != NULL)

{

pD = S.pop();

cout << pD->info;

delete pD;

}

pTrav = pTrav->pNext;

CNode\* ptD;

ct = 0;

while (pTrav != NULL)

{

while (ptD != NULL)

{

ct++;

ptD = ptD->pDown;

if (ct > posC + 1)

{

break;

}

}

if (ct == posC + 1)

{

break;

}

pTrav = pTrav->pNext;

}

//pTrav is now at the col with same num of nodes and now display

CNode\* pS = pTrav;

while (pS != NULL)

{

pnn = new CNode;

pnn->info = pS->info;

pnn->pNext = pS->pNext = NULL;

S.push(pnn);

pS = pS->pDown;

}

CNode\* pD;

while (S.pHead != NULL)

{

pD = S.pop();

cout << pD->info;

delete pD;

}

/////////////////////////////////////////////////////////////////

/\*My method\*/

int ct=0;

int half = (N + 1) / 2;

half++;

//if N=9, half=5 and then half=6 bcuz after the 5th node, we decrease the num of nodes

for (int i = 1; i <= N; i++)

{

pnn = new CNode;

cout << "enter info pnn \n";

cin >> pnn->info;

pnn->pNext = NULL;

pnn->pDown = NULL;

L.Attach(pnn);

if (i < half)

{

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

{

pnn1 = new CNode;

cout << "enter info pnn1 \n";

cin >> pnn1->info;

pnn1->pNext = NULL;

pnn1->pDown = NULL;

if (pnn1->pDown == NULL)

{

pnn->pDown = pnn1;

pnn->pDown->pNext = pnn1;

}

else

{

pnn->pDown->pNext->pDown = pnn1;

pnn->pDown->pNext = pnn1;

}

}

}

else

{

ct+=2; //each time it loses two of the nodes it was supposed to create

for (int j = 0; j < i-ct; j--)

{

pnn1 = new CNode;

cout << "enter info pnn1 \n";

cin >> pnn1->info;

pnn1->pNext = NULL;

pnn1->pDown = NULL;

if (pnn1->pDown == NULL)

{

pnn->pDown = pnn1;

pnn->pDown->pNext = pnn1;

}

else

{

pnn->pDown->pNext->pDown = pnn1;

pnn->pDown->pNext = pnn1;

}

}

}

}

/\*WITHOUT STACK\*/

cout << "enter posC \n";

cin >> posC;

CNode\* pTrav = L.pHead;

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

{

pTrav = pTrav->pNext;

}

CNode\* ptD = pTrav->pDown; //this will traverse down the pTrav(column wanted)

int stop = posC;

while (stop>0)

{

ct = 0;

ptD = pTrav->pDown;

while (ct != stop)

{

ptD = ptD->pDown;

ct++;

}

cout << ptD->info <<" ";

stop--;

}

//to look for second col

pTrav = pTrav->pNext;

ptD=pTrav->pDown;

ct = 0;

while (pTrav != NULL)

{

while (ptD != NULL)

{

ct++;

ptD = ptD->pDown;

if (ct > posC + 1)

{

break;

}

}

if (ct == posC + 1)

{

break;

}

pTrav = pTrav->pNext;

}

//pTrav is now at the col with same num of nodes and now display

ptD = pTrav->pDown; //this will traverse down the pTrav(second col)

stop = posC;

while (stop > 0)

{

ct = 0;

ptD = pTrav->pDown;

while (ct != stop)

{

ptD = ptD->pDown;

ct++;

}

cout << ptD->info << " ";

stop--;

}

}