考前複習&加分~

Double Linking List:

利用Week6的Queue來做修改可以簡單很多，

此題分三部份，一部份4分整題12分。

在之前題目中我們實作了，

isFull()、isEmpty()、EnQueue()、DeQueue()、Clear()，

而EnQueue()、DeQueue()是〝新增〞在最後一個以及〝去除〞第一個node，

**第一部份(四分)：**

使用上次的Queue，改成雙向鏈結。

\*\*\*請注意備註部份的改動\*\*\*

**第二部份(四分)及第三部份(四分)：**

在此希望新增兩個Function，

分別為

InsertNode();

// 新增到第N個的，第N個的前一個為NULL則回傳失敗，否則回傳1

//包含輸入<1的情況回傳失敗

// EX：我Queue只有兩個Node，我卻輸入insert(4)則回傳失敗

DeleteNode();

// 刪除第N個，第N個為NULL則回傳失敗，否則回傳1

//包含輸入<1的情況回傳失敗

**並將insert新增為功能3，delete新增為功能4，clear改成功能5。**

(1)Add node

(2)Remove node

(3)Insert node to behind nth

(4)Delete nth node

(5)clear

**備註：**

除新增此兩個Function外，還多了一個限制，在之前作業中並沒有硬性規定Queue的第一個Node是從哪裡開始，這次統一從〝Headnode.next〞為第一個Node。

EX：

若我們有一個Queue為： 5 4 3

則對應的Node為Headnode ->next ->next ->next

front rear

可能有些人會注意到，Headnode.value會沒被使用到，那我們利用他的value來紀錄此queue的長度，可以使isFull()和isEmpty()簡單許多。

(以前一個範例為例，Headnode.value即為3。)

小提醒:

-建議先把之前的改成dbllinklist的想法後再攥寫新的function比較不會有

忘記修改的地方。

-insert與delete可能會有跟enqueue和dequeue重複的情況，可以利用已經

實作完的dequeue和enqueue。

-很多人常常會在node已經等於NULL的時候，再去使用它的

next,pre,value導致程式當掉，若執行錯誤可往這個方去檢查。

-insertnode(0),deletenode(0)是不一樣的喔，記得做處理。

.h檔

**typedef** **struct** node{

**struct** node \*pre;

**struct** node \*next;

**int** value;

}node\_t,\*p\_node\_t;

**typedef** **struct** queue{

p\_node\_t front; // First

p\_node\_t rear; // Final

p\_node\_t headNode; // Queue的領導node，不被刪除。

**int** size; // size是指queue中最多node數量, size 訂5個

}queue\_t,\*p\_queue\_t;

**p\_queue\_t** **Create**();

// Initial Queue,Return a Queue.

**int** **isEmpty**(p\_queue\_t quu);

// queue 空的回傳1

**int** **isFull**(p\_queue\_t quu);

// queue 滿的回傳1

**int** **deQueue**(p\_queue\_t quu);

// 移除一個node，並回傳是否移除成功，成功回傳1

**int** **enQueue**(p\_queue\_t quu,**int** value);

// 添加一個node，並回傳是否添加成功，成功回傳1

**void** **clear**(p\_queue\_t quu);

// 清空queue

**int** **InsertNode**(p\_queue\_t quu,**int** nth,**int** value);

// 新增到第N個，第N個pre為NULL則回傳失敗，否則回傳1

**int** **DeleteNode**(p\_queue\_t quu,**int** nth);

// 刪除第N個，第N個為NULL則回傳失敗，否足回傳1

// 兩者可用quu->headNode->value判斷失敗情況

輸入： 輸出：

1 6 6

1 7 6 7

1 8 6 7 8

1 9 6 7 8 9

1 10 6 7 8 9 10

1 11 FULL

4 3 6 7 9 10

3 3 50 6 7 50 9 10

2 7 50 9 10

2 50 9 10

3 5 20 Error Node

4 5 Error Node

4 -1 Error Node

4 3 50 9

4 0 Error Node

3 1 23 23 50 9

3 4 10 23 50 9 10

5 clear

2 Empty