資料結構與進階程式設計（107-2）
作業五

作業設計：楊其恆
國立臺灣大學資訊管理學系

繳交作業時，請將第一題與第二題的答案以中文或英文作答後，以 PDF 檔上傳到 NTU COOL；***不接受紙本繳交***；第三題請至 PDOGS（`http://pdogs.ntu.im/judge/`）上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。這份作業的截止時間是 ***2019 年 5 月 6 日星期一凌晨 1:00***。在你開始前，請閱讀課本的第 8、9 章[1] 不接受遲交。

第一題

(20 分) 多重選擇題，每題有 n 個選項，答錯 k 個選項者，得該題 $\frac{n-2k}{n}$ 的題分。

1. 給定一個可能包含 0 至 n 個元素的 list `aList`，各方法的定義同課本中之定義，下列敘述何者必然正確?

   (A) `aList.getLength() = (aList.remove(i)).getLength() + 1`

   (B) `aList.getLength() = (aList.insert(i, 0).getLength() - 1`

   (C) `(aList.remove(i)).insert(i, 0) = aList`

   (D) `(aList.setEntry(i, x)).getEntry(i) = x`

   (E) `(aList.clear()).getLength() = 0`

2. 請考慮課本中對於 ADT list 的二種 implementation 方法，令 n 為 list 中的元素數量，下列敘述何者正確?

   (A) 使用 array-based implementation，在最糟情形下 insert 的複雜度為 $O(1)$

   (B) 使用 link-based implementation，在最糟情形下 insert 的複雜度為 $O(n)$

   (C) 使用 array-based impelmentation，在最糟情形下 getEntry 的複雜度為 $O(1)$

   (D) 使用 link-based implementation，在最糟情形下 getEntry 的複雜度為 $O(n)$

   (E) 使用 recursive link-based implementation，會使 insert 在最糟情形下的複雜度較一般的 link-based implementation 低。

第二題

（20 分）考慮一個只具有以下方法的 list，並假設我們對於 list 的存取只限於使用以下三個函數，無法得知內部的 private data。

- `insert(newPosition, newEntry)`

[1] 課本是 Carrano and Henry 著的 *Data Abstraction and Problem Solving with C++: Walls and Mirrors* 第六版。

- `remove(position)`
- `getLength()`

現在希望使用這個 list 取代常用的 array 或 linked list 來實作一個 stack，請依序回答以下數題

(a) 請以程式碼說明如何利用以上的方法，來實作 list 的`getEntry(position)` 方法。

(b) 請說明 stack 的成員除了 list 之外，應該還有哪些，以及如何使用這些成員來儲存 stack 中的資料。

(c) 請以 C++ 程式碼實作 stack 的 top() 方法，若有需要可以文字輔助說明。

(d) 請以 C++ 程式碼實作 stack 的 push(elmnt) 與 pop() 方法，若有需要可以文字輔助說明。

第三題

（60 分）在 Python 中，list 是一個基本且常用的資料結構，它的概念與 ADT list 幾乎相同，且因為可以直接取得某個特定元素的值，在大多時候甚至會被當成 python 中的陣列使用。Python list 可以允許放入各種不同資料型態的資料，無論是整數、文字，或是另一個物件，都可以被放入同一個 list 之中。在本題中，要請大家實作一個簡易版的 python list，並使用 python 的語法對 list 物件進行操作，而為了降低麻煩，我們只考慮在 list 中放入整數的情形。常見的 python list method 有以下幾個

1. aList.append(elmnt): 將 elmnt 這個整數放入 aList 之中，並且放在最後面。
2. aList.insert(pos, elmnt): 將 elmnt 這個整數放在 list 的第 pos 個 index 位置，如果 pos 大於或等於 list 的長度，請直接將 elmnt 放在 list 的最後。
3. aList.remove(elmnt): 若 elmnt 有在 aList 之中出現，則將第一個出現的移除。
4. aList.index(elmnt): 取得 elmnt 在 aList 之中第一次出現的 index。
5. aList.count(elmnt): 取得 elmnt 在 aList 之中出現的次數。
6. aList.extend(anotherList): 將 anotherList 接到 aList 之後。舉例來說，如果 aList 的內容是 [1,2,3]，而 anotherList 的內容是 [5,6,7]，則執行完此指令後，aList 的內容會變成 [1,2,3,5,6,7]，anotherList 則不受影響。
7. aList.reverse(): 將 aList 之中的元素順序全部顛倒。
8. aList.sort(): 依照大小順序將 list 之中的元素由小到大排好。

另外，python 中的 list 會以`aList = list()` 或`aList = []` 的方式宣告並初始化這個物件為空的 list。在本題的輸入資料中，我們一律使用前者作為初始化之指令。就如同 C++ 中的`List* aList = new List()`，代表相同的意義。只是在 python 中回傳的不會是指標，而是一個空的物件。

輸入輸出格式

系統會提供一共 30 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有若干行，其中每一行包含一個 python 指令，python 的指令可能有上述的多種，此外對於以下的幾種，請根據對應的指令輸出應有的結果

1. aList: 請依序印出 aList 之中的內容，以逗號分隔其中的各個整數，並在前後印出中括號。因此若 aList 之中沒有內容，請直接印出`[]`。
2. aList.count(elmnt): 請計算 aList 之中 elmnt 出現的次數，並印出這個次數。
3. aList.index(elmnt): 請印出 elmnt 在 aList 之中的 index，若此元素並沒有在 aList 中出現，請印出 -1。

對於每一個需要印出東西的指令，請於印出結果後換行。在本題中，以指令宣告的 list 物件不會超過 30 個，但指令的數量沒有上限。測試資料的內容分布如下表所示：

測試資料	append	insert	remove	index	count	extend	reverse	sort
1-10	Y	Y						
11-15	Y	Y	Y					
16-20	Y	Y	Y	Y	Y			
21-25	Y	Y	Y	Y	Y	Y		
26-30	Y	Y	Y	Y	Y	Y	Y	Y

舉例來說，若輸入為

```
a = list()
a
a.append(5)
a.append(10)
a.append(15)
a.append(20)
a.append(25)
a
a.insert(3, 17)
a.remove(20)
a.append(15)
a
b = list()
b.append(15)
b.append(25)
b.append(50)
b.remove(17)
b
b.index(17)
a.extend(b)
```

```
a
a.count(15)
a.index(15)
a.reverse()
a
a.sort()
a
```

則輸出應該為

```
[]
[5,10,15,20,25]
[5,10,15,17,25,15]
[15,25,50]
-1
[5,10,15,17,25,15,15,25,50]
3
2
[50,25,15,15,25,17,15,10,5]
[5,10,15,15,15,17,25,25,50]
```

你上傳的原始碼裡應該包含什麼

你的.cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你***只能使用 link-based implementation***。

評分原則

這一題的分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。