國立台灣海洋大學資訊工程系 C++ 程式設計 期末考參考答案

姓名：_____

系級：_____

學號：_____

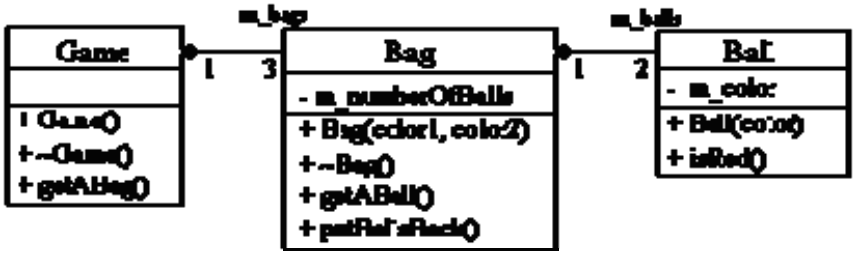
102/06/18

考試時間：**10:00 – 12:00**

請看清楚題目問什麼，針對重點回答，總分有 *120*，請看清楚每一題所佔的分數

考試規則：1. 不可以翻閱參考書、作業及程式
2. 不得使用任何形式的電腦 (包含計算機)
3. 不得左顧右盼、不得交談、不得交換任何資料、試卷題目有任何疑問請舉手發問 (看不懂題目不見得是你的問題，有可能是中英文名詞的問題)、最重要的是隔壁的答案可能比你的還差，白卷通常比錯得和隔壁一模一樣要好
4. 提早繳卷同學請直接離開教室，不得逗留喧嘩
5. 違反上述任何一點之同學以作弊論，一律送請校方處理
6. 繳卷時請繳交***簽名過之試題卷及答案卷***

1. [65] 在課堂中我們解釋過一個 3-bags 的程式，這個程式中有三種物件 Game, Bag, Ball，分工合作來完成一個機率實驗，類別圖和原本的界面設計如下圖:



在實習中我們也曾經修改過這個程式，讓 Game 裡面有四個袋子，每個 Bag 裡有三個球，還結合兩個骰子進行下面修改過的機率實驗:

> 1. 請由四個袋子中任意挑選一袋
> 2. 請由袋子中選一顆球出來，如果是紅球的話，把球放回去，重新執行步驟 1
> 3. 如果在步驟 2 中挑到白球，請隨機挑一顆骰子丟，如果是奇數的話，不做任何事，如果是偶數的話再挑一顆球，把球留在外面
> 4. 最後再由袋子裡挑一顆球,請計算這樣子挑到白球的機率

a) [10] 假設我們希望每一個 Game 都有自己專屬的骰子，每個骰子有自己的機率分佈，請修改上面的類別圖，加入骰子的類別，骰子的界面包括建構元，解構元，以及 randomThrow(), 並且畫出骰子類別和其它類別的關係?

Sol:

b) [55] 假設我們現在想要設計另外一種機率實驗，這個實驗裡有的袋子裡有三個球: 類別 ThreeBallBag，有的袋子裡有四個球: 類別 FourBallBag，袋子的種類是在建構 Game 物件時以各 1/2 的機率隨機決定的，請設計抽象類別 Bag 來使得 Game 裡面的 m_bags 變成一個異質指標容器 (需要定義純粹虛擬函式)，並且運用繼承架構來設計 Bag, ThreeBallBag，和 FourBallBag 三個類別之間的關係

i. [5] 請畫出這個設計的類別圖(包括 Game, Bag, ThreeBallBag, FourBallBag，和 Ball)

Sol:

ii. [10] 請寫出 Bag 抽象類別，以及 ThreeBallBag, FourBallBag 類別的 class 定義

Sol:

iii. [5] 請寫出 Game 類別的 class 定義

Sol:

iv. [10] 請寫出 Game::Game()建構元 (如果是四個球的袋子，球的顏色是兩白兩紅，如果是三個球的袋子，球的顏色是兩紅一白)

Sol:

v. [10] 根據下列實驗步驟

> 1. 請由四個袋子中任意挑選一袋
> 2. 請由袋子中選一顆球出來，如果是紅球的話，把球放回去，重新執行步驟 1
> 3. 如果在步驟 2 中挑到白球，請由剩餘的三個袋子中隨機挑選一個袋子
> 4. 由袋子裡連續挑兩顆球,請計算在此步驟中挑到一紅一白球的機率

在 main() 函式中寫出重複執行 10000 次實驗時機率估計的程式

Sol:

vi. [5] 請說明 v. 中何者為多型指標，在此程式中其特性為何?

Sol:

vii. [10] 請問 v. 中呼叫 Bag::getABall() 是動態繫結還是靜態繫結? (什麼語法造成的?) 以此例而言請說明這兩種繫結的差異為何?

Sol:

2. [10] 在作業二中，如果操作者輸入的時間先後順序不對，希望你的程式檢查出來並且適當地處理，請問你認為應該用 if 敘述來嘗試由錯誤狀況中自動回復，還是應該使用 assert 敘述停止程式執行? 你的判斷準則是什麼?

Sol:

在作業二中，我們為了快速地模擬保養廠的運作，在開發這個應用程式的時候設計由測試程式的人在每一個動作之前輸入時間，如果輸入的時間先後順序不對，例如後輸入的時間比先前輸入的時間還早，程式應該要檢查出來，避免後續的邏輯判斷錯誤；任何時候你應該要釐清兩個不同的角色，一個是軟體開發完了以後真正操作這個軟體的人，另一個是軟體開發者/測試者的角色, if 敘述是軟體邏輯的一部份，應該針對以後操作這個軟體的人來設計, assert 則一定是針對開發者/測試者的，所以你在評估的時候不論你平常喜歡用哪一種，評估的準則中要考量到執行者的角色；以這個問題來說，輸入時間的錯誤應該不是正常運作的保養廠管理程式會遇見的，實際上線的程式中，時間值應該要改成讀取系統時間，應該是一個遞增的

數值，所以應該不需要以 if 寫在系統裡，程式開發過程中檢查各種假設，只需要用 assert 寫出即可，最後上線的版本中前處理器可以自動地去除這些敘述。

3. [10] 在作業二中，如果你定義了“物料”類別，每一個物料類別的物件都代表一種物料，由 Parts.txt 檔案中可以讀出很多種物料，這些物料你會用一個容器物件把他們記錄下來，例如用標準函式庫中的 vector，你有沒有發現你的程式裡會有好多地方需要搜尋這個 vector 裡面的所有物料，例如計算每一個維修項目的費用時，針對所需要更換的每一個物料你需要搜尋 vector 一次；在列印維修單據時，可能你又需要針對所需要更換的每一個物料搜尋 vector 一次；這種出現重複的程式碼的現象其實就引出一個“是否需要多一層封裝”的問題：是否需要設計一個“物料表格”或是“物料資料庫”的類別把前面使用的 vector 包裝起來，然後把搜尋的功能改成是“物料表格”的介面，並且實作搜尋某一物料 (例如物料編號是 a31f) 的功能?

a. [5] 請運用 vector 定義一個“物料表格”類別?

Sol:

```
class PartTable
{
public:
    PartTable(istream &is);
    Part *findPart(string id) const;
Private:
    vector<Part> m_parts;
};
```

b. [5] 請實作搜尋某一物料的界面?

Sol:

```
Part *PartTable::findPart(string id) const
{
    for (int i=0; i<m.parts.size(); i++)
        if (m_parts[i].IDEquals(id))
            return &m_parts[i];
    return 0;
}
```

4. [10] 就“程式碼重用 (code reuse)”的角度來看，請比較運用“繼承(inheritance)”和 運用“組合(composition)/聚合(aggregation)”來建構物件階層有什麼相同的地方? (請分別就繼承以及組合/聚合 *舉例* 說明) 繼承比組合/聚合多了什麼? 請問 C++ private inheritance 的語法是上述哪一種的實作?

Sol:

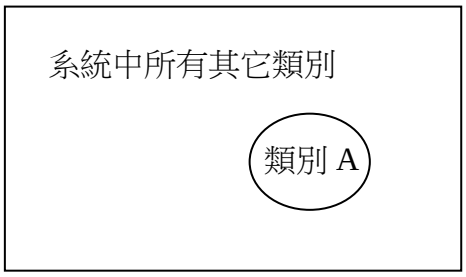
不管是“繼承”或是“組合/聚合”，都是運用現有的類別已經規劃好的資料和成員函式的功能來建構新的類別的功能，例如第一題的 Bag 和 Ball 物件之間的關係就是“組合/聚合”，Bag 不需要把 Ball 所有的功能都寫到它自己的類別中，只需要記下對應的 Ball 物件或是 Ball 物件的指標，遇見需要由 Ball 物件的功能來完成的狀況時，委託 Ball 物件來完成就可以了。第一題中 ThreeBallBag 和 Bag 的關係就是“繼承”，在設計 ThreeBallBag 時, Bag 裡已經

完成的功能當然也不需要重新再做一遍了，或者說 ThreeBallBag 和 FourBallBag 兩個類別中共同的資料和共同的程式都可以寫到 Bag 類別中

```
class Bag
{
public:
    virtual Ball *getABall();
};
class ThreeBallBag: public Bag
{
public:
    virtual Ball *getABall();
private:
    int m_numberOfBalls;
    Ball *m_balls[3];
};
```

繼承

聚合



“繼承”比“組合/聚合”多出來的特性是“父類別物件的介面也是子類別物件的介面”;“繼承”是 IS-A 的關係,“組合/聚合”則是 HAS-A 的關係; 如上圖如果我們針對某一類別把程式分為兩部份: 其它類別與類別 A, 繼承類別 A 時是一種 factoring 的機制來重用所有其它類別中原本使用類別 A 物件的程式碼, 子類別的物件直接就可以看成是父類別的物件在系統中運作, 也就是 old code call new code 的精神; 反之“組合/聚合”雖然是一種運用 library function 的機制來重用類別 A 的程式碼, 產生的新物件卻沒有辦法直接取代類別 A 的物件在系統中運作, 不像繼承一樣直接重用其它類別的程式碼。

C++ 的 private inheritance 是一種“組合/聚合”機制的實作, 並不是實作物件導向中的“繼承”概念。

5. [25] 下圖中 Derived 類別繼承 Base 類別, Base 類別的 m_dataPtr 是一個整數指標, 在 Base::Base(int numInt) 建構元中配置可以存放 numInt 個整數的陣列給 Base 類別的物件使用:

```
01. class Base {
02. public:
03.     Base();
04.     Base(int numInt);
05.     virtual ~Base();
06. private:
07.     int m_numInt;
08.     int *m_dataPtr;
09. };
10. class Derived: public Base {
11. public:
12.     …
13. private:
14.     double m_figures[3];
15. };
```

a. [5] 請撰寫 Base 類別的拷貝建構元?

Sol:

```
--- Base.h ---
class Base {
```

```
public:
    Base(const Base &src);
    …
};
--- Base.cpp ---
Base::Base(const Base &src): m_numInt(src.m_numInt)
{
    m_dataPtr = new int[src.m_numInt];
    for (int i=0; i<src.m_numInt; i++)
        m_dataPtr[i] = src.m_dataPtr[i];
}
```

b. [5] 請撰寫 Derived 類別的拷貝建構元?

Sol:

```
--- Derived.h ---
class Derived {
public:
    Derived(const Derived &src);
    …
};
--- Derived.cpp ---
Derived::Derived(const Derived &src): Base(src)
{
    for (int i=0; i<3; i++)
        m_figures[i] = src.m_figures[i];
}
```

c. [5] 請問如果 Derived 類別不撰寫拷貝建構元, 編譯器會不會自動合成一個拷貝建構元? 這個建構元能夠正確運作嗎? (請說明可以或是不行的原因)

Sol:

會, 可以正確運作, 編譯器合成的拷貝建構元會運用 Base::Base(const Base&) 來初始化 Base 父類別的物件, 可以順利拷貝父類別 Base 物件, Derived 類別內沒有配置記憶體, 所以合成的拷貝建構元可以拷貝 Derived 類別內所有資料

d. [10] 假設你已經撰寫了 Base 類別的設定運算子 (assignment operator), 請撰寫 Derived 類別的設定運算子?

Sol:

```
--- Derived.h ---
class Derived {
public:
    Derived& operator=(const Derived &rhs);
    …
};
--- Derived.cpp ---
```

```
Derived& Derived::operator=(const Derived &rhs)
{
	if (&rhs == this) return *this;
	for (int i=0; i<3; i++)
		m_figures[i] = src.m_figures[i];
	Base::operator=(rhs);
	return *this;
}
```