程式設計（113-1）
期末考

題目設計：孔令傑
國立臺灣大學資訊管理學系

本次考試為手寫考試，考題會以紙本提供給大家，考生只能用筆在官方提供的答案紙上作答。考試是 closed-book 的，考生唯一能帶和查閱的是一張考生自己準備的 A4 紙，這張紙的兩面都可以由考生在考前任意記載任何資訊，但這張紙上不能浮貼其它物體。考生在考試期間不可以跟其他人直接或間接討論題目和答案。考試時若有問題，可以舉手問助教；作答時可以不依順序，但要清楚標明題號；可以用任何筆，包含鉛筆，只要清楚就行；筆跡要清晰，如果助教看不懂紙上的文字，就只能扣分。

這次的考試時間為 ***2024 年 12 月 17 日下午 14 點 20 分***至當日 ***下午 17 點 20 分***，共 180 分鐘。本次考試總分共 100 分，學生必須作答前兩題，並且在第三、四題中選一題作答。如果有學生在第三、四題都有作答，則只有其中一題會被納入計分。

第一題（必答）

（30 分）有一個連鎖速食餐廳準備開發自動點餐系統（就是大家現在在麥當勞、摩斯等等常看到的那種）。該餐廳的餐點很單純，只有薯條、飲料、漢堡；薯條只分小份、中份、大份；飲料只有汽水和紅茶，各有小杯、中杯、大杯。目前小份、中份、大份的薯條價格各為 30、35 和 40 元；小杯、中杯、大杯的汽水價格各為 25、35 和 45 元；小杯、中杯、大杯的紅茶價格各為 20、30 和 40 元。

在該自動點餐系統中，有人寫了如下兩個類別，這樣就可以在使用者新增一個品項到購物車裡時新增一個相對應的物件：

```
#include<iostream>
using namespace std;

const int FRIES_SIZE_CNT = 3;
const int DRINK_TYPE_CNT = 2;
const int DRINK_SIZE_CNT = 3;

class Fries
{
private:
  static const int prices[FRIES_SIZE_CNT];
  int size; // 0: small, 1: medium, 2: large
```

```
public:
  Fries(int size) : size(size) {}
  int getSize() { return this->size; }
  void setSize(int size) { this->size = size; }
  int getPrice() { return Fries::prices[this->size]; }
};

const int Fries::prices[FRIES_SIZE_CNT] = {30, 35, 40};

class Drink
{
private:
  static const int prices[DRINK_TYPE_CNT][DRINK_SIZE_CNT];
  int type; // 0: soda, 1: black tea
  int size; // 0: small, 1: medium, 2: large
public:
  Drink(int type, int size) : type(type), size(size) {}
  int getType() { return this->type; }
  void setType(int type) { this->type = type; }
  int getSize() { return this->size; }
  void setSize(int size) { this->size = size; }
  int getPrice() { return Drink::prices[this->type][this->size]; }
};

const int Drink::prices[DRINK_TYPE_CNT][DRINK_SIZE_CNT]
  = {{25, 35, 45}, {20, 30, 40}};
```

已知前述的程式碼是完全正確的。

根據前述資訊，請獨立地回答以下各小題（亦即回答每一小題時，請假設其他小題並不存在）。

(a)（7 分）前述的兩個類別，如果改寫成結構（`struct`），會需要做哪些必要的調整？這樣改寫又有什麼好處或壞處？為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(b)（7 分）下面是一個使用前述類別定義的主程式：

```
int main()
{
```

```
  Fries f(1);
  Drink d(0, 2);
  cout << f.getPrice() + d.getPrice();
  return 0;
}
```

這個主程式是否可以正確編譯出執行檔？若可以，執行時是否會 run-time error？若不會 run-time error，會印出什麼？不論你的答案為何，請簡要地說明你的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(c)（8 分）在 `Drink` 中，是否有任何成員函數應該被設為 static 的成員函數？如果有，請指出來並且簡要地說明原因（不超過 100 個字）；如果沒有，請寫「沒有」且不需要說明原因。此外，是否有任何成員函數應該被設為 constant 的成員函數？如果有，請指出來並且簡要地說明原因（不超過 100 個字）；如果沒有，請寫「沒有」且不需要說明原因。

(d)（8 分）有人認為應該要把 `Fries` 和 `Drink` 的 static 成員變數 `prices` 改成 instance 成員變數，這樣比較好開發、比較好維護、將來比較好擴充，程式執行也比較不耗資源。對於這個看法，你認同嗎？不論你的答案為何，請簡要地說明你的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

第二題（必答）

（35 分）承第一題（但請忽略該題的各小題，只要關心該題題幹中的資訊即可），該速食餐廳也有漢堡，包含牛肉漢堡、鱈魚堡和蛋堡[1]，每個漢堡都有兩片麵包、零或一片生菜、零或一片番茄、一片牛肉或一片鱈魚或一片煎蛋。

有人寫了下面這個類別：

```
class Burger
{
private:
  int type; // 0: beef, 1: fish, 2: egg
  bool hasLettuce;
  bool hasTomato;
public:
  Burger(int type, bool hasLettuce, bool hasTomato)
    : type(type), hasLettuce(hasLettuce), hasTomato(hasTomato) {}
  int getType() { return this->type; }
  void setType(int type) { this->type = type; }
```

[1]世界上有些人是吃素的，重要！

```
  bool getHasLettuce() { return this->hasLettuce; }
  void setHasLettuce(bool hasLettuce)
    { this->hasLettuce = hasLettuce; }
  bool getHasTomato() { return this->hasTomato; }
  void setHasTomato(bool hasTomato) { this->hasTomato = hasTomato; }
  int getPrice();
};
```

已知前述的程式碼除了 `getPrice()` 還沒定義以外，都是完全正確的。

根據前述資訊，請依序回答以下各小題。

(a)（7 分）假設公司改變政策，允許客人點餐時在漢堡中放不只一片生菜（當然要加錢），請修改 `Burger` 的定義以允許這個彈性。你只要說明原本的哪裡該被修改、要被修改成什麼樣子即可；你不用重抄一次那些不用被修改的程式碼，也不用說明修改的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(b)（7 分）延續前一小題，該餐廳的漢堡計價方式如下：牛肉漢堡、鱈魚堡和蛋堡原則上各是 90 元、80 元和 60 元，如果客人不要生菜則一律減 5 元，要多一片則加 5 元；如果客人不要番茄則一律減 10 元，要多一片則加 10 元。請模仿第一題的程式碼，去擴充 `Burger` 的定義，以實作能回傳一個漢堡價格的 `getPrice()` 函數。實作時請包含 function header；你不用重抄一次那些不用被修改的程式碼，也不用加入文字或註解說明你的程式碼。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(c)（7 分）延續前一小題。有人認為應該要幫 `Burger` 寫 default constructor，方便建立物件。對於這個看法，你認同嗎？不論你的答案為何，請簡要地說明你的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(d)（7 分）延續第 (b) 小題。有人認為應該要幫 `Burger` 寫 copy constructor、destructor、assignment operator，以免未來出現不好除錯的 run-time error。對於這個看法，你認同嗎？不論你的答案為何，請簡要地說明你的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(e)（7 分）延續第 (b) 小題。有人認為應該要幫牛肉漢堡、鱈魚堡、蛋堡各寫一個類別，它們都各自繼承 `Burger`，然後現在 `Burger` 中的 `type` 就可以拿掉，許多成員變數和成員函數在三個子類別裡也不用重新定義，也不需要有 static 的陣列，只要在三個子類別中各有一個 static 的整數去記錄該種漢堡的基本價格即可。對於這個看法，你認同嗎？不論你的答案為何，請簡要地說明你的原因。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

第三題（和第四題擇一回答）

（35 分）承第二題，但請忽略該題的各小題，只要關心題幹並假設 `getPrice()` 已經被實作完成了即可。

該餐廳現在要推出套餐，一個套餐一定包含一個漢堡、一杯飲料和一份薯條，且該套餐的價格就是這三個品項的價格加總[2]。顯然我們應該幫「套餐」寫一個類別。

(a)（10 分）有人建議幫套餐建立一個類別如下（constructor 就是一對一地設定成員變數的值，這邊為了簡單起見就直接忽略，請當作已經寫好了）：

```
class MealSet
{
private:
  int burgerType;
  bool hasLettuce;
  bool hasTomato;
  int drinkType;
  int drinkSize;
  int friesSize;
public:
  MealSet(bType, bHL, bHT, dType, dSize, fSize)
  {
    // naive initialization
  }
  getPrice(); // how to implement MealSet::getPrice()?
};
```

簡言之，當客人在自動點餐機選擇他要買一個套餐時，他自然會選他要怎樣的漢堡、飲料和薯條，那就把這些參數拿來建立這個套餐物件即可。在這個架構下，請實作 `MealSet` 的 `getPrice()` 函數。由於 `Burger`、`Drink` 和 `Fries` 的 `getPrice()` 都寫好了，且我們顯然不希望當這些品項個別的價格改變時還要改 `MealSet` 的 `getPrice()`，請務必呼叫個品項的 `getPrice()` 來實作。你不可以修改 `MealSet` 的架構；實作時請包含 function header；你不用重抄一次那些不用被修改的程式碼，也不用加入文字或註解說明你的程式碼。

(b)（10 分）關於前一小題的 `MealSet` 架構，撇開排版、變數命名不談，它還是有在架構上不妥的地方，例如會導致效率不彰、容易 run-time error、未來不易擴充、容易造成前後不一致等問題的其中若干項或其他問題。請列舉兩項你認為最嚴重的、最能說服別人修

[2]實務上當然可以加總後打個折，但在本題這不是重點，因此我們就不打折了。

改原始架構的問題，並且說明原因。為了避免大家煩惱要說明到怎樣的詳細程度，本題兩個議題合計限（大約）400 字。

在回答時，請不要嘗試幫本題的程式做更多它目前沒有要做的事。舉例來說，它目前沒有 getter 和 setter，也沒有使用 template、exception、operator overloading 等等，就請假設它確實不需要這些，因此回答時請不要包含這些。你也可以假設未來一個套餐永遠都會是一個漢堡、一杯飲料和一份薯條。總之，請就眼前的程式及其功能作答即可。

在批改本題時，如果你列出的問題沒有涵蓋到最主要的問題，你也會被扣一點分數；如果答案中有列出其實沒有問題的地方，也會被扣一點分數。但不論如何，即使沒有完全正確，批改時都會根據內容給部份分數。

(c)（15 分）如果你認為前一小題的架構不好，請設計一個新架構。你不用實作，只要說明你的新架構長什麼樣子即可，包含 (1) 成員變數有哪些、(2) constructor 要怎麼運作、(3) `getPrice()` 要怎麼運作，以及 (4) 是否需要為了你的新架構而有在原本的舊架構上不需要的東西。你的架構未必是完美的，亦即可能會有前一小題的架構沒有的缺點，但它應該要避免你列出的不妥之處。為了避免大家煩惱要說明到怎樣的詳細程度，本題的新架構說明限（大約）400 字。

第四題（和第三題擇一回答）

（35 分）承第二題，但請忽略該題的各小題，只要關心題幹並假設 `getPrice()` 已經被實作完成了即可。

該餐廳的自動點餐系統中要有個購物車功能，讓使用者可以自由增減購物車中的品項，例如依序新增一個一般的漢堡、一個不要生菜的鱈魚堡、一個要兩片生菜的蛋堡、一份大份薯條、一杯中杯汽水、一杯小杯紅茶，之類的。假設客人一次買複數個品項時要付的總價格就是所有品項的單價總和[3]。購物車內的品項數最多只能有 20 個。

(a)（10 分）顯然我們應該幫「購物車」寫一個類別。有人建議，我們可以先建立一個叫 `ITEM_CNT_MAX` 的常數，代表購物車中最多可以放幾個品項；接著幫購物車建立三個陣列 `burgers`、`drinks`、`fries` 當 instance variable，分別放或指向被消費者加入購物車的漢堡、飲料和薯條物件；並且幫購物車建立四個整數 `burgerCnt`、 `drinkCnt`、`friesCnt` 和 `itemCnt`，分別代表購物車中的漢堡、飲料、薯條品項數和總品項數。

這個設計顯然是有缺點的，因此有人建議你善用繼承，讓 instance variable 只需要有一個整數 `itemCnt` 和一個陣列 `items`。請說明 (1) 這是否有可能被做到，(2) 如果可以，要怎麼做（你只要說明即可，例如說明讓哪個類別繼承哪個類別、繼承時是否需要定義新的成員、是否需要定義新類別等等；你不需要實作），以及 (3) 如果可以，新的架構下不

[3]實務上當然可以加總後打個折，但在本題這不是重點，因此我們就不打折了。

同種類的品項為什麼可以被放進同一個陣列。為了避免大家在本題花太多時間，本題限 400 字，請大家挑重點寫即可。

(b)（10 分）承前一小題，如果用了繼承，顯然多型也應該要被使用。舉例來說，購物車應該也有一個 `getPrice()` 函數，會回傳購物車中所有品項的總價。這個函數會把 `items` 陣列裡的前 `itemCnt` 個元素的 `getPrice()` 都各呼叫一次，並且把這些回傳的結果加總後回傳。請說明當你已經建立了前一小題的繼承關係後，你該做些什麼才能正確地實作本小題說的 `getPrice()`，包含 (1) 在親類別（parent class）中要做的、(2) 在子類別（child class）中要做的，以及 (3) 在購物車的 `getPrice()` 中要做的。為了避免大家煩惱要敘述得多細，本題限 400 字，請大家挑重點寫即可。

(c)（5 分）你要幫這個購物車類別實作運算子多載，並且多載索引運算子（indexing operator）`[]`，目標是如果有個購物車物件 `shoppingCart`，則 `shoppingCart[i]` 會回傳購物車中第 $i+1$ 個品項。假設你的整個程式中都不會有常數物件。在這個實作中，請問你的回傳值型態應該是什麼？是個物件、物件參照、指標，還是指標參照？請簡要說明理由，不用實作。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(d)（5 分）你的同伴幫這個購物車類別寫了一個 `addItem()` 的成員函數，可以傳入正確的資訊後試著讓購物車中多一個品項，如果成功就回傳 `true`，反之則不增加品項並且回傳 `false`；這個函數具體怎麼運作不是重點。現在，你要修改這個 `addItem()` 函數，以確保在購物車已經滿了的情況下如果繼續新增品項，呼叫 `addItem()` 的地方必須做些處理（例如在指定的 TXT 檔中新增一筆資料以留下紀錄之類的）。請注意呼叫 `addItem()` 的程式是又另一個同伴寫的，不是你寫的，你當然也不能把他的程式碼拿來改，你只能改 `addItem()`；他遇到這個狀況時要做什麼也由他決定，你只要確定他有做些什麼就好。請說明你的修改，並簡要說明理由；只要說明，不用實作。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。

(e)（5 分）針對 `Burger` 這個類別，有一位長官 A 提議價格不要是整數，而是使用浮點數 `float` 或 `double`，這樣將來這個自動點餐機的程式才能賣到日常價格有小數點後位數的國家；另一位長官 B 雖然同意長官 A 的看法，但則認為價格只會是整數的國家，程式還是應該維持以整數為資料型別，這樣程式運行時比較節省空間。請問，有沒有辦法修改 `Burger` 的定義，讓同一個 `Burger` 定義在不同國家完全不用修改，就能讓建立漢堡物件的程式自行決定要以浮點數還是整數做為價格的資料型別？如果有，請簡要說明你要使用什麼作法，以及該作法在修改 `Burger` 方面大致上有哪些步驟，修改完之後在建立漢堡物件的程式中又要怎麼指定資料型別？請說明你的修改及使用方式，並簡要說明理由；只要說明，不用實作。為了避免大家煩惱要敘述得多細，本題限 200 字，請大家挑重點寫即可。