計算機概論 資訊一甲 一零四學年度 第二學期 期末考試
學號：　　姓名：Solutions

1. (20 points) 假設有一個 C++ 程式語言的有理數 (rational number) 類別 **class** Rational，這個類別物件變數有下列宣告的例子：
   (a) Rational r(4, 8); (b) Rational s(r); (c) Rational t(-10); (d) Rational u(6.25);
   寫出 **class** Rational 建構函式 (constructors) 的**定義介面**。
   (a) Rational(**cons int** = 0, **const int** = 1);
   (b) Rational(**const** Rational&);
   (c) Rational(**const short** &); (也可由 (a) 的 default value 宣告。)
   (d) Rational(**const double** &);

2. (10 points) 假設一個 C++ 程式語言中的佇列 (queue) 使用雙向連結表 (double-linked list) 實作，其節點 (node) 類別 **class** Node 和佇列 (queue) 類別 **class** Queue 的介面定義如下：

```
class Node {
  private:
    int data;
    Node *prev;
    Node *next; };
```

```
class Queue {
  private:
    Node *head;
    Node *tail;
```

```
public:
    Queue();
    ~Queue();
    void enqueue(int);
    int  dequeue();
    int  first(); };
```

   寫出 **class** Queue 解構函式 (destructor) 的**程式碼** (提示：當一個佇列被解構時，這個佇列的所有節點必須被移除；你可使用其他成員函式 (member functions)。)

```
Queue:: ~Queue() {
  while (head != NULL) this->dequeue(); }
```

3. (20 points) 回答下列各小題：
   (a) 考慮第二題節點類別 **class** Node 和佇列類別 **class** Queue 的定義，將其改寫為**類別模板** (class template)。
   (b) 假設有一個學生類別 **class** Student，我們要寫一個 C++ 語言的程式模擬學生排隊買票的情境，這個買票的隊伍在程式中叫做 ticket_line。使用 (a) 小題的模板，如何表示這個 ticket_line?
   (c) 如果一個學生類別的物件 (student object) 為 40 個位元組 (bytes)，在程式執行某一時刻 ticket_line 的隊伍有 120 個學生，此時**物件變數** ticket_line 有多少個位元組? **整個隊伍**有多少個位元組?
   (a)
   節點類別模板：

```
template <class T>
class Node {
  private:
    T data;
    Node *prev;
    Node *next; };
```

   佇列類別模板：

```
template <class T>
class Queue {
  private:
    Node<T> *head;
    Node<T> *tail;
  public:
    Queue();
    ~Queue();
    void enqueue(T);
    T  dequeue();
    T  first(); };
```

   (b) 物件變數 ticket_line 的宣告為：

```
Queue<Student> tickent_line;
```

   (c) 物件變數 ticket_line 只有兩個節點指標 (node pointer)，每個指標是 4 個位元組，所以 ticket_line 共有 8 個位元組。整個隊伍有 120 位學生，每個學生使用一個節點表示，每個節點有一筆學生資料 40

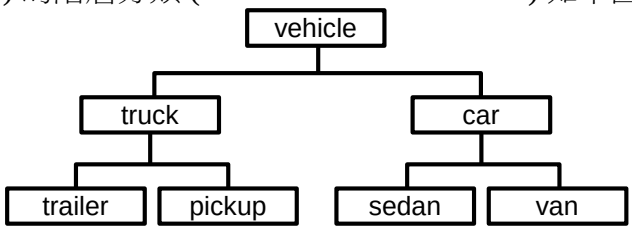
個位元組和兩個指標 8 個位元組。因此一個節點有 48 個位元組；120 位學生有 120 個節點，為 5760 位元組；整個隊伍包含學生的記憶體空間和 ticket_line 的記憶體空間，所以總共是 5760+8 = 5768 個位元組。

4. (30 points) 假設有一個 C++ 程式語言的整數堆疊 (integer stack) 類別 **class** Stack，變數 **int** m 是一個整數以及變數 Stack s, t 是堆疊物件變數。如果 **class** Stack 的應用程式 (application program) 有多負載運算 (overloaded operators) 的算術式 (arithmetic expression)，寫出下列多負載運算的**定義介面** (提示：請勿寫出各多負載運算的實作；)：

(a) –s // 回傳和 s 反向 (reversed) 的堆疊；
(b) s+m // 將整數 m 放入堆疊 s 的頭部，並回傳處理結果的堆疊；
(c) m+s // 將整數 m 放入堆疊 s 的底部，並回傳處理結果的堆疊；
(d) s-m // 將堆疊 s 中所有整數 m 的元素移除，並回傳移除元素後的堆疊；
(e) s+t // 將 s 和 t 兩個堆疊的元素放在一起，s 的元素在下，t 的元素在上；並回傳該堆疊；
(f) s-t // 若堆疊 t 的元素出現在堆疊 s 中，則將該元素從 s 中移除；並回傳移除元素後的堆疊；

(a) Stack operator-(**void**) **const**;
(b) Stack operator+(**const int** &) **const**;
(c) **friend** Stack operator+(**const int** &, **const** Stack &) **const**;
(d) Stack operator-(**const int** &) **const**;
(e) Stack operator+(**const** Stack &) **const**;
(f) Stack operator-(**const** Stack &) **const**;

5. (20 points) 假設車輛 (vehicle) 的階層分類 (hierarchical classification) 如下圖所示：



(a) 此分類將車輛 (vehicle) 分為貨車 (truck) 和客車 (car)；所有車輛都需紀錄其車輛牌照 (license, 格式為 *AAA-dddd*，*A* 是英文字母，*d* 是數字)、車廠 (manufacture, 已知車廠如 Ford, General Motor, Toyota, Nissan, Honda, BMW, Mercedes Benz, Yulon)、出廠年份 (year)。
(b) 貨車又分為聯結車 (trailer) 和載貨車 (pickup)；貨車要記錄車輛長度 (length) 和寬度 (width)；
(c) 聯結車要記錄其用途 (usage)，如貨櫃車、油罐車、露營車等；載貨車要記錄載重量 (weight，單位：公噸)。
(d) 客車又分為轎車 (sedan) 和廂型車 (van)。客車要記錄車輛排氣量 (displacement)；
(e) 轎車要記錄多少門數 (door)；廂型車要記錄最多乘坐人數 (passenger)。

依照此階層分類架構圖寫出 C++ 程式語言的**繼承類別** (inheritance classes)，只寫出**私有和保護資料成員** (private and protected data member)。(提示：你可使用題目中的英文字作為類別或資料成員的名稱。)

```
class Vehicle {
  private:
    char * license;
    char * manufacture;
     int   year;
  public:
     …;
};

class Truck: public Vehicle {
  private:
    char * usage;
  public:
     …;
};

class Trailer: public Truck {
  private:
    float  weight;
};

class Pickup: public Truck {
  private:
    float  weight;
};
```

```
  private:
    float    length;
    float    width;
  public:
     …;
};

class Car: public Vehicle {
  private:
    float    displacement;
  public:
     …;
};

  public:
     …;
};

class Sedan: public Car {
  private:
    int      door;
  public:
     …;
};

class Van: public Car {
  private:
    int      passenger;
  public:
     …;
};
```