資料結構與程式設計

(Data Structure and Programming)

101 學年上學期複選必修課程 901 31900

Homework #1.2 (Due: 11:00pm, Oct 10, 2012)

Department:____________________________ Grade:___________

Id:________________________ Name:_________________________

1. Getting familiar with multi-file project and Makefile.

(a) Write a header file "hw1.2.p1a.h" that contains the following class:

```
class P1a
{
public:
   P1a() {}
   P1a(const string& s) : _str(s) {}
   void assign(const string&);
   void print() const;
   P1a& append(const P1a&);

private:
   int    _dummy;
   string _str;
};
```

Write a C++ file "hw1.2.p1a.cpp" that implements `class P1a`'s member functions:

(i) The member function "`void assign(const string& s)`" assigns the "`string s`" to the data member "`_str`" by the "=" operator.

(ii) The member function "`void P1a::print() const`" prints out the data member "`_str`" followed by an "`endl`".

(iii) The member function "`P1a& append(const P1a& p)`" appends the string "`p._str`" to the data member "`_str`" and return a "reference variable of the calling object".

Note that the C++ file “hw1.2.p1a.main.cpp” is included in the homework files.

(b) Implement a C++ file “hw1.2.p1b.cpp” that does the following:

(i) A function[1] “`void printSize()`” to print out the size of `class P1a` as:

```
The size of class P1a is XX.
```

(Replace XX with the actual size)

(ii) A function “`void printStaticArraySize()`” that declares an array of `class P1a`:

```
   P1a arr1b_1[5];
```

and prints out the memory addresses of “`arr1b_1[i]’s`” in hex as:

```
============================
Addresses of arr1b_1[5] are:
&arr1b_1[0]: 0x123456ab
&arr1b_1[1]: .....
&arr1b_1[2]: .....
&arr1b_1[3]: .....
&arr1b_1[4]: .....
```

(Replace 0x123456ab and “.....” with actual addresses)

(iii) A function “`void printDynamicArraySize()`” that declares an array of `class P1a`:

```
   P1a *arr1b_2 = new P1a[5];
```

, prints out the memory addresses of “`arr1b_2[i]’s`” in hex as:

```
============================
Addresses of arr1b_2[5] are:
&arr1b_2[0]: 0x123456ab
&arr1b_2[1]: .....
&arr1b_2[2]: .....
&arr1b_2[3]: .....
&arr1b_2[4]: .....
```

and prints out the memory address of the variable “`arr1b_2`” in hex as:

```
&arr1b_2: .....
```

[1] When not explicitly specified, a “function” means the file-scope function, not a member function.

(Replace 0x123456ab and "....." with actual addresses)

(iv) A function "`void printPointerArraySize()`" that declares an array of `class P1a` pointers:

```
    P1a **arr1b_3 = new P1a *[5];
    for (size_t i = 0; i < 5; ++i)
       arr1b_3[i] = new P1a;
```

, prints out the memory addresses of "`arr1b_3[i]'s`" in hex as:

```
============================
Addresses of arr1b_3[5] are:
&arr1b_3[0]: 0x123456ab
&arr1b_3[1]: .....
&arr1b_3[2]: .....
&arr1b_3[3]: .....
&arr1b_3[4]: .....
```

, prints out the contents of "`arr1b_3[i]'s`" in hex as:

```
Contents of arr1b_3[5] are:
arr1b_3[0]: .....
arr1b_3[1]: .....
arr1b_3[2]: .....
arr1b_3[3]: .....
arr1b_3[4]: .....
```

and prints out the memory address of the variable "`arr1b_3`" in hex as:

```
&arr1b_3: .....
```

(Replace 0x123456ab and "....." with actual addresses/values)

Implement a C++ file "hw1.2.p1b.main.cpp" that includes an "`int main()`" that calls all the functions in "hw1.2.p1b.cpp" (with the order as specified above).

(c) Compose a "Makefile" that is able to compile various executables:

(i) Type "make p1a" to compile the source codes in (a). The generated executable should be called "hw1.2.p1a". Execute it and direct the outputs to a file "hw1.2.p1a.out".

(ii) Type "make p1b" to compile the source codes in (b). The generated executable should be called "hw1.2.p1b". Execute it and direct the outputs to a file "hw1.2.p1b.out".

(iii) Type “make bug” to compile the necessary source codes including the C++ file “hw1.2.p1.bug.cpp”. The generated executable should be called “hw1.2.p1.bug”.

(iv) The default make target should be “p1a” (i.e. by typing “make”).

(d) Execute “hw1.2.p1.bug” and you should see the program crashes. Use the debugger “gdb” or “ddd” for debugging. Describe how you debug and explain the source/reason of the bug in the file “hw1.2.p1d.out”.

2. This is a problem taken from “National Problem Solving Contest on Internet (NPSC) 網際網路程式設計全國大賽” 2009. We will use this problem to guide you how to write an OOP program.

[Problem Description]

瘋狂博士 X 是一個很龜毛的人，他有很多奇怪的原則，例如：不穿有兩顆釦子的衣服、坐下來的時候不可以朝向東方、下雨天不能坐公車、喝黑咖啡要用深藍色的馬克杯，如果是加糖的咖啡一定要用白色的咖啡杯裝著，而且加糖的步驟一定要是先加 1 塊方糖、攪拌 2 下、加入 5 毫升的鮮奶、再攪拌 7 下，最後看他當天的心情決定要補加多少砂糖

瑞特是瘋狂博士 X 的鄰居，每個星期天下午會到瘋狂博士 X 家幫他整理家務。瑞特的工作包括整理瘋狂博士 X 的小型圖書館，最近瘋狂博士 X 買回家的書似乎有越來越多的趨勢，他希望有人可以幫他寫一個有下列功能的程式：輸入瑞特手邊的書籍資料，程式會自動輸出這些書籍的排序。

值得注意的是，在瘋狂博士 X 的心中，AEIOU 五個母音的順序是 OAUIE，而且 X 是 26 個字母之首；所以對於瘋狂博士 X 而言，英文字母的順序如下：XOBCDAFGHUJKLMNIPQRSTEVWYZ。

書籍擺放方式的規則：

i. 先按照作者的名字排序，

ii. 作者相同的書籍再按照書籍問世的年份排序，

iii. 如果作者、出版年份都相同則按照書名排序。

千萬不要忘記這裡的順序是按照瘋狂博士 X 心目中的英文字母順序排列！

[Problem Inputs]

總共只有一筆測試資料。

第一行有一個整數 N，代表總共有幾本書籍資料，至多 100 本書。

接下來共有 N 行，每一行描述一本書籍，格式為：「書籍作者, 書籍名稱 (出版年份)」，書籍作者和書籍名稱中間由一個逗點及一個空白字元分開。

「書籍作者」包括大小寫英文字母、半形句點或空白字元，至多 30 個字元。

「書籍名稱」包括大小寫英文字母或空白字元，至多 100 個字元。

「出版年份」必定為四位數字。

本題的輸入測試檔案皆會符合上述要求，不用考慮格式錯誤的問題，也不用考慮會有多出來的空格 (如逗號之前多一個空格之類的)。

[Sample Inputs]

```
12
Erle Stanley Gardner, Drifting Down the Delta (1969)
Agatha Christie, Five Little Pigs (1942)
Erle Stanley Gardner, Host With the Big Hat (1969)
J. R. R. Tolkien, Leaf by Niggle (1945)
Agatha Christie, The Body in the Library (1942)
Erle Stanley Gardner, The Case of the Murderers Bride (1969)
Erle Stanley Gardner, The Case of the Stuttering Bishop (1937)
J. R. R. Tolkien, The Hobbit (1937)
J. R. R. Tolkien, The Lay of Aotrou and Itroun (1945)
Agatha Christie, The Moving Finger (1942)
Crazy Doctor X, AAA (2009)
Crazy Doctor X, XXX (2009)
```

[Problem Outputs]

排序後的書單，按照瘋狂博士 X 心目中的英文字母順序排列與要求。排序時同一字母的大小寫視為相等，但輸出資料的大小寫必須和輸入資料一致。空白字元的順序較英文字母為先。半形句點的順序在空白字元和英文字母之間。

總共有 N 行輸出，格式為「書籍作者, 書籍名稱 (出版年份)」，與輸入格式相同。

[Sample Outputs]

```
Crazy Doctor X, XXX (2009)
Crazy Doctor X, AAA (2009)
Agatha Christie, Five Little Pigs (1942)
Agatha Christie, The Body in the Library (1942)
Agatha Christie, The Moving Finger (1942)
J. R. R. Tolkien, The Hobbit (1937)
J. R. R. Tolkien, Leaf by Niggle (1945)
J. R. R. Tolkien, The Lay of Aotrou and Itroun (1945)
Erle Stanley Gardner, The Case of the Stuttering Bishop (1937)
Erle Stanley Gardner, Drifting Down the Delta (1969)
Erle Stanley Gardner, Host With the Big Hat (1969)
Erle Stanley Gardner, The Case of the Murderers Bride (1969)
```

Complete the TODO’s of the following sub-problems.

(a) (Operator overloading) The program "hw1.2.p2a.cpp" defines a class "xStr" that takes a char string and converts it to a new upper-case string with the alphabetic order in Crazy Doc's mind. For example, the string "XO" will be converted to "AB", and "dog" will be converted to "EBH". This is to facilitate the comparison of strings using the existing string comparison functions.

A sample input/output is:

```
Type Ctrl-C to terminate the program...

First string  : Test
Crazy Doc's   : UVTU
==============================
Enter a string: dog
Crazy Doc's   : EBH
>> dog is smaller than Test
>> dog is not equal to Test
==============================
Enter a string: abc
Crazy Doc's   : FCD
>> abc is not smaller than dog
>> abc is not equal to dog
==============================
Enter a string: ha123
Crazy Doc's   : IF123
>> ha123 is not smaller than abc
>> ha123 is not equal to abc
==============================
```

(b) ("string" manipulation) The program "hw1.2.p2b.cpp" defines a class "Book" that records the book information with the author (_author) and book (_book) names in class xStr, and the year. You should overload the operator '<' by following Crazy Doc's rules in sorting books. The input file "hw1.2.p2b.in" is provided, and your generated output file "hw1.2.p2b.out" should match the sample outputs above.

3. For the following "SelectionSort" function,

```
void selectionSort(vector<int>& array)
{
   for (size_t i = 0, n = array.size(); i < n - 1; ++i) {
      size_t pivot = i;
      for (size_t j = i+1; j < n; ++j) {
         if (!compare(array[pivot], array[j]))
            pivot = j;
      }
      if (pivot != i)
         mySwap(array[pivot], array[i]);
   }
}
```

(a) Implement the "main()" function so that the program can ask the user to input a sequence of numbers and store them in an array. The above function "SelectionSort()" will then be called by "main()" to sort the numbers in ascending order. Please also implement the functions “`compare()`” and “`mySwap()`” as called in “`SelectionSort()`”. However, please DO NOT change the code in “`SelectionSort()`”. The following is a sample output:

```
How many numbers? 5
234
132
532
123
52
Before sort:
234 132 532 123 52
After sort:
52 123 132 234 532
```

For ease of grading, put your code in a single file called “hw1.2.p3a.cpp”.

(b) Change the “`compare()`” function into a “*functional object*” argument for the “`selectionSort()`” function. That is, the function prototype will be:

```
void selectionSort
(vector<int>& array, const Compare& compare);
```

Please create another two functional object classes, “`Less`” and “`Greater`”, which inherit the class “`Compare`”. Therefore, in “`main()`”, if we pass in a functional object of type “`Less`” (i.e. calling “`selectionSort(`*arr*`, Less())`”), the numbers will be sorted in ascending order. On the contrary, the function object of type “`Greater`” will sort the numbers in descending order.

Please make the base functional object class “`Compare`” a virtual class. That is, make its “`operator ()`” a pure virtual function.

Sort the numbers in “`main()`” in both ascending and descending order. A sample output is as follows:

```
How many numbers? 5
234
132
532
123
52
Before sort:
234 132 532 123 52
Ascending sort:
52 123 132 234 532
Descending sort:
532 234 132 123 52
```

For ease of grading, put your code in a single file called “hw1.2.p3b.cpp”.

(c) Modify the functions into template functions. That is, the “selectionSort” will not only be able to sort integers, but any data type as long as its “operator <” and “operator >” are defined.

In other words, the function prototype of “selectionSort()” will be:

```
template <class T>
void selectionSort
(vector<T>& array, const Compare<T>& compare);
```

In your “main()”, sort a sequence of strings in ascending order, and a sequence of doubles in descending order. A sample output is as follows:

```
How many strings? 5
hello
ric
what
is
up
Before sort:
hello ric what is up
Ascending sort:
hello is ric up what

How many doubles? 6
1.2
3.8
-23.43
-0.02
8.88
10
Before sort:
1.2 3.8 -23.43 -0.02 8.88 10
Descending sort:
10 8.88 3.8 1.2 -0.02 -23.43
```

For ease of grading, put your code in a single file called “hw1.2.p3c.cpp”.

Notes: Please pay attention to the homework rules on the website. Failure to abide by the rules may result in deduction in homework points.