CS2310 Computer Programming
2018/19 Semester B
Department of Computer Science
City University of Hong Kong
Assignment Three
Due Date: 4 May 2018 (Friday) 23:00

All questions should be submitted to PASS.
This assignment consists of 9 pages, including this cover page. It contains 4 questions named as Q1 to Q4.

In Q1-Q4, DO NOT INCLUDE any library other than <iostream>, <iomanip>, <fstream> and/or <cmath> in your solution. Each solution of Q1-Q4 using any function/facility in other libraries will receive ZERO mark.

Solution Submission Instruction:

- You are required to submit a complete C++ program in one file to PASS.
- You can test the program with VS2015, and submit as many times as you want before the deadline. We will use the latest version of your submission for grading.
- In each question above on PASS, only a small set of test cases are available to you for testing. We will use a more comprehensive set of test cases for grading your solution. Therefore, passing all test casts do NOT mean you will get 100% correct in our grading tests. Try to test your program thoroughly before final submission. Happy coding! ☺

Grading:

- Solution correctness (90%) and style (10%).
- **Correctness**: We will use PASS to grade for correctness, basing on how many test cases are correct with your program. If your program cannot be compiled and tested on PASS, you will get **ZERO** mark.
- **Style**: We will check your source code for indentation, variable naming, comments, etc.
- Some differences for question 2, which will be described in the *Note* part of question 2.

Plagiarism Check:
The course will use multiple code plagiarism detection software to check every solution the whole class submits. These software tools include the following two tools. Students are suggested to keep important immediate versions and paper draft of their solutions in case that students are asked to show evidences of self-effort and originality in developing their solutions of each question. All suspicious cases will be forwarded to the CS Departmental Disciplinary Committee for further actions.

- PASS: https://pass3.cs.cityu.edu.hk/index.jsp and
- MOSS: https://theory.stanford.edu/~aiken/moss.

NEVER place your solution in the common network directory used in the tutorial sessions.

Q1. Complete the class definition (10%)

Based on the declaration for class **Complex** as following, write a program to:

1. Complete the definition of the member functions declared;
2. Call these functions in *main* function. Follow the same format of inputs and outputs with examples.

Note: if the imaginary part of the complex number is zero, you just need to display the real part when output the complex number.

```
Class Complex
{
Private:
double real;
double imag;

Public:
// initialize the real and imaginary parts by a and b respectively
void Initialize(double a, double b);

// print the complex number on the screen
void show();

// compute and return the power of the current complex number
Complex power();

// compute and return the multiplication of two complex numbers
Complex multip(Complex A);
};
```

Example:

```
Enter two complex numbers:
1 -1 3 4
The complex numbers you entered are:
1 - 1i
3 + 4i
The power of 1 - 1i is:
0 - 2i
The power of 3 + 4i is:
-7 + 24i
The result of (1 - 1i) * (3 + 4i) is:
7 + 1i
```

Q2. Pointer and Dynamic Array (30%)

A dice has 6 faces, each with one unique value: 1, 2, 3, 4, 5, and 6. The face value of a dice means the value showing upward. Write a program to finish tasks as the followings:

1. Find the number of occurrences of the sum of face values of the two dices.
2. On top of 1, sort the number of occurrences in descending order.
3. On top of 2, rather than showing the exact occurrence count of each sum of face values, output the letters A,B,C, … so that A represents the highest occurrence count, B represents the 2nd highest occurrence count, and so on.
4. The outputs should have the following format:

```
1.
```

```
The number of occurrences:
1 occurrence(s) of the sum 2
2 occurrence(s) of the sum 3
3 occurrence(s) of the sum 4
4 occurrence(s) of the sum 5
5 occurrence(s) of the sum 6
6 occurrence(s) of the sum 7
5 occurrence(s) of the sum 8
4 occurrence(s) of the sum 9
3 occurrence(s) of the sum 10
2 occurrence(s) of the sum 11
1 occurrence(s) of the sum 12
```

```
2.
```

```
The sorted number of occurrences:
6 occurrence(s) of the sum 7
5 occurrence(s) of the sum 6
5 occurrence(s) of the sum 8
4 occurrence(s) of the sum 5
4 occurrence(s) of the sum 9
3 occurrence(s) of the sum 4
3 occurrence(s) of the sum 10
2 occurrence(s) of the sum 3
2 occurrence(s) of the sum 11
1 occurrence(s) of the sum 2
1 occurrence(s) of the sum 12
```

```
3.
```

```
The sorted number of occurrences in letters:
F occurrence(s) of the sum 2
E occurrence(s) of the sum 3
D occurrence(s) of the sum 4
C occurrence(s) of the sum 5
B occurrence(s) of the sum 6
A occurrence(s) of the sum 7
B occurrence(s) of the sum 8
C occurrence(s) of the sum 9
D occurrence(s) of the sum 10
E occurrence(s) of the sum 11
F occurrence(s) of the sum 12
```

Requirements:

1. Every array in your program must be a dynamic array.
2. Array elements can only be accessed via pointers.

3. Every dynamic array must be deleted before the program terminates.
   Hint: when deleting a 1D array, use delete: `delete[] sum;` when deleting a 2D array, you need to delete all the rows of the 2D array first, and then the 2D array.

Note:

1. This question will be divided into 3 sub-questions and one test case for each sub-question in PASS system. Your outputs should be exactly the same as shown above.
2. We will check your source code to see if your program meets the requirements. If not, you will not be graded as correctness even if your outputs is correct.

Q3. Library borrowing system (30%)

In this question, you are required to design a simple library borrowing system and finish tasks as the followings:

1. Accept the information about the books, including book name, data, state and subject.
2. Assign the books into different subject categories according to the input.
   For simplicity, assume:
   - There are only **three** subject categories in this small library.
   - Each book only can be assigned into **one** subject.
3. List all the books and their information under each subject category respectively. The books listed in each subject should be sorted by date (newest to oldest).
4. Accept the name of the book that the reader want to borrow. If the book is unavailable, print a message and ask the reader to input again.

Requirements:

1. The concept of *Library* is implemented as a C++ class.
   - Each *Library* object owns three Subject objects, named Art, Science, and History.
     Hint: the use of an array to keep three Subject objects will make your program significantly more complex than keeping them as three separate variables.
2. The concept of *Subject* is implemented as a C++ class.
   - Each *Subject* contains its own list of *Books* and the count of *Books*.
     Hint: here you can create an object array to store all the *Books*.
3. The concept of *Book* is implemented as a C++ class.
   - Each *Book* object has its own Name (csting), Date (int) and State (bool, 1: available, 0: not available).
4. **All data members of each object can only be accessed by the object itself.**

More description about the above three classes

Ideas on the data members of the **Subject** class. [The class may contain other data members]	
count	The number of Books
list	An array of Books

Ideas on the data members of the **Library** class. [The class may contain other data members]	
Name	The name of this Library object
Art	The first Subject owned by the Library
Science	The second Subject owned by the Library
History	The third Subject owned by the Library

Ideas on the data members of the **Book** class. [The class may contain other data members]	
Name	The name of this book object
Date	The date of the book object
State	The state of the book object

Hint: You can assume that: 1) the number of *Books* is at least 1 and at most 100; 2) any cstring consists of at most 10 characters; 3) all user inputs are valid.

Example:

```
Input the number of books:
9
Input the information of books (name, date, state, subject):
book1 1990 0 Art
book2 1995 1 History
book3 1992 1 History
book4 2007 0 Science
book5 2017 1 Art
book6 1997 1 History
book7 2001 0 Science
book8 2014 1 Science
book9 2000 1 Art
Books in the library:
Art:
book1 1990 0
book9 2000 1
book5 2017 1
Science:
book7 2001 0
book4 2007 0
book8 2014 1
History:
book3 1992 1
book2 1995 1
book6 1997 1
Input the name of the book you want to borrow:
book1
Sorry, the book is not available. Try again:
book3
Succeed!
```

Q4. **Cstring (30%)**

C++ has a set of functions implementing operations on strings inherited from the C language. These functions support various operations, such as copying, concatenation, searching, etc.

Write a program to implement some of these operations using your own code.

Requirements:

Declare your own functions as the followings:

1. **char * mystrncat (char * destination, const char * source, int num);**
   Append the first *num* characters of *source* to *destination*. If the length of the string in *source* is less than *num*, only the content up to the terminating null-character is copied. Assume that the *destination* is large enough to contain the concatenated resulting string.
2. **char * mystrtok (char * str , const char * delimiters);**
   A sequence of calls to this function split *str* into tokens, which are sequences of contiguous characters separated by any of the characters that are part of *delimiters.* For more details about this function, you can refer to the website below:
   http://www.cplusplus.com/reference/cstring/strtok/
   Hint: you might use **local static variable** to save the remainder of the original string after the delimiter.
3. **char * mystrcmp (const char * str1, const char * str2, char * result);**
   Compares the string *str1* to the string *str2*. *result* is separated into two parts: 1) **num1** denotes the number of characters that contained in both strings and located at the same position; 2) **num2** denotes the number of characters contained in both strings but located at different positions. The format of *result* is:

A	num1	B	num2

   For example, if *str1* = `"c45d"`, *str2* = `"s45c"`, then *result* = `"A2B1"`.
   Assume that str1 and str2 contain the same number of characters, and each string has all unique characters.

Call your own functions in *main* function. Follow the same format of inputs and outputs with examples.

Examples:

1.

```
Input the destination:
cs2310 computer
Input the source:
programming
Input the number of characters to be appended:
20
The concatenated resulting string is:
cs2310 computerprogramming
```

2.

Input a sentence:

cs2310 computer programming 2017/18

Input the delimiters:

/

The tokens are:

cs2310

computer

programming

2017

18

3.

Input the first string:

2da53gf4

Input the second string:

2f4rxgdl

The result of comparison is:

A2B3