Assignment #2, Advanced Programming Lab. 2016 Spring Semester

Due date: 2016/05/06, 23:59:59

1. [FILE I/O] Write a program that reads a file named ‘phone_book.txt’ and retrieve data on a screen by query. In addition, write the output into ‘output.txt’ file. In the input file, ‘phone_book.txt’, name and phone number are separated by a comma. Multiple same names can be stored but all the phone numbers are unique. If the same names are queried, you must show all the entries. (Do NOT use any string functions.)

<text file example> - ‘phone_book.txt’ 김철수,010-1234-1234 김영희,010-2222-3333 박성현,010-5555-6666 김철수,010-7777-8888
Ex_1) Select (1:Name, 2:Phone): 1 Name: 김철수 - ‘output.txt’ 김철수 010-1234-1234 김철수 010-7777-8888
Ex_2) Select (1:Name, 2:Phone): 2 Phone: 010-1234-1234 - ‘output.txt’ 김철수 010-1234-1234

2. [FILE/IO] Write a program that reads a text file including multiple words and then prints all the words in ascending order. In a text file, English words are stored in random order and a word consists of no more than 15 characters. Also, words are separated by a single space or a carriage return. Note that the sorting is not case-sensitive. (Do NOT use any string functions.)
   The output should be as follows.

< text file example > - word.txt Car banana Apple Brush crayon Air	Ex) Text file: word.txt Air Apple banana Brush Car crayon

3. [Pointer, Dynamic Allocation] Write a program that reads a 2-D array (N×M) and then prints all the elements in ascending order. The numbers in the array should range from 0 to 99 in integer type. The rows and columns may be different.

Ex1)	Ex2)
Input size: 3 3 Input array: 17 22 1 2 31 5 34 4 14 Output 1 2 4 5 14 17 22 31 34	Input size: 2 4 Input array 2 4 7 3 1 9 6 8 Output 1 2 3 4 6 7 8 9

4. [Pointer, Dynamic Allocation] Write a program that reads a matrix A of N×M size and a matrix B of M×O and then prints the multiplication result of the matrix A and B. The values of the matrix are integers. The program should handle all the possible exceptions of matrix calculation.

Ex1)	Ex2)
Matrix A size: 2 2 Matrix B size: 2 3 A: 1 2 3 4 B: 5 6 7 8 9 10 Multiplication result 21 24 27 47 54 61	Matrix A size: 2 3 Matrix B size: 2 3 Incomputable 3 != 2

5. [String] Write a program that reads a sentence and locations of two words for switching and then prints the switched sentence. The sentence must consist of more than two words and a single period at the end of the sentence. Notice that the length of the two words may be different. You are supposed to give two numbers indicating the locations of words to be switched. (Do NOT use any string functions.)

Ex)
Input sentence: I like apples but I hate bananas. Switch locations: 3 7 Switched sentence I like bananas but I hate apples.

6. [String] As you know, a URL(Uniform Resource Locator) consists of a web site path, data names, and values (e.g., www.kw.ac.kr/index.php?student_id=2015722001&grade=2&department=computer_engineering). Write a program that reads a URL and break it into a path and data passed by the GET method then prints them separately. You must write your own function (my_strtok) to tokenize the URL. The prototype of 'my_strtok' function is as follows:

char* ***my_strtok***(char* str)

This function must operate similar to the built-in 'strtok' function which separates a string by several delimiters. Unlike the 'strtok' function, 'my_strtok' function has two default delimiters ('?', and '&'). The parameter 'str' is the pointer indicating the string that will be separated by 'my_strtok' function. This function returns the address of the first token of 'str' in each function call. After the first token extraction, use NULL as the function parameter to tokenize the remaining URL string. Each data of the URL must have a one '='. The separated tokens can be used for data searching. The program should handle all the possible exceptions of input. You are NOT allowed to use any string functions.

```
Ex)
Input URL: www.kw.ac.kr/index.php?student_id=2015722001&grade=2&department=computer_engineering
www.kw.ac.kr/index.php
student_id=2015722001
grade=2
department=computer_engineering

Search data: department
department is computer_engineering.
```

7. [Class] Implement a 'MyCar' class which can save car information. MyCar class has private member variables to store model, color, and plate number. Implement member functions to display all member variables on the screen. Create three objects and reads data from keyboard. Class prototype is described below.

```
class MyCar
{
    private:
        char* m_Model;
        char* m_Color;
        int m_Number;

    public:
        MyCar(char* m_Model, char* m_Color, int m_Number);
        ~MyCar();

        char* getModel();
        char* getColor();
        int getNumber();
};
```

8. [Class, FILE I/O] In ‘books.txt’ file, book title and quantity are stored. Implement the Load, Add, Edit, Print, and Save functions for this file. You must implement ‘MyBook’ class which holds a book information. The title and quantity are divided by ‘/’.
   - Load function: Read book information from ‘books.txt’ and fill ‘MyBook’ class array.
   - Add function: Add a book title and quantity to the list.
   - Edit function: Find a book title and edit the book quantity.
   - Print function: Print all the information of the stored books.
   - Save function: Save ‘MyBook’ class array to ‘books.txt’ file.

   The output should be as follows. (Do NOT use any string functions.)

<text file example> - ‘books.txt’
A structured approach using C++/12
Introduction to Probability & Statistics/20

Ex)
1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 1

1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 4

[Book list]
A structured approach using C++/12
Introduction to Probability & Statistics/20

1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 2
Book title: 선형대수
Quantity: 20

1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 3
Book title: Introduction to Probability & Statistics
Quantity: 10

1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 5

1. Load 2. Add 3. Edit 4. Print 5. Save 6. Quit
Input number: 6

<result text file example> - ‘books.txt’
A structured approach using C++/12
Introduction to Probability & Statistics/10
선형대수/20

9. [Linked List] Implement a student information management system with the linked list. The class 'Student' saves students' name and ID number. The class 'StudentList' keeps the head node pointer and handles all list operations, and the Student class works as a node in the list.

   The StudentList has the following member functions:
   - Insert: Add a student object to the last position of the linked-list.
   - Search: Search all the stored nodes by name or name & student ID number. Display all the member variables (name and ID number) of the retrieved nodes on the screen. (If there are the same names, display all retrieved nodes)
   - Sort: Sort the list in ascending order of the name. (If there are the same names, sort them with the student ID number)
   - PrintAll: Display all student information stored in the list on the screen. (Output sequence is equal to the list order)

   Each class prototype is described below.
   (Names can be the same but student ID numbers are different. Suppose the names do not have any spaces.)

```
class StudentList
{
    private:
        Student* s_pHead;

    public:
        StudentList();
        ~StudentList();

        void Insert(char* name, int number);
        void Search(char* name);
        void Search(char* name, int number);
        void Sort();
        void PrintAll();
};
```

```
class Student
{
    private:
        char s_Name[15];
        int s_Number;
        Student* s_pNext;

    public:
        Student();
        ~Student();

        void setName(char* name);
        void setNumber(int number);
        void setNext(Student* pNext);

        char* getName();
        int getNumber();
        Student* getNext();
};
```

10. [Linked List] Generate a linked-list of palindrome words and implement a function to delete a single node. Palindrome words are obtained from the 'palindrome.txt' file. The linked-list should be implemented in ascending order of the words. Input an integer value, n from the user. The n-th node in the list can be removed. Display all the nodes stored in the list after deletion.
    (Palindrome is a word that consists of the same characters in backwards and forwards. ex> bob, dad, radar, cammac)

<text file example> - 'palindrome.txt' Given a **bob** application area for which we want to design an embedded system, we must determine **radar** specific goals for **deified** the project and estimate **dad** their feasibility.
Ex) list: bob -> dad -> deified -> radar Delete?: 3 list: bob -> dad -> radar Delete?: