**OOP345**

**Assignment**

**Part A**

**Program # 1**

You are to create a database of books that are stored using a vector. Keep track of the author, title and publication date of each book. Your program should have a main menu that allows the user to select from the following:

1. Add a book’s author, title, and date
2. Print an alphabetical list of the books sorted by author
3. Quit

You must use a class to hold the data for each book. This class must hold three string fields, one to hold the author’s name, one for the publication date, and another to hold the book’s title. Store the entire database of books in a vector where each vector element is a book class object.

To sort the data, use the generic sort function from the <algorithm> library. Note that this requires you to define the <operator to compare two objects of type book so that the author field from the two books are compared.

***Note: you are required to create three files, Book.h, Book.cpp and TestProgram.cpp.***

A sample of the input/output behavior might look as follows. Your I/O need not look identical, this is just to give you an idea of the functionality.

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 1**

Enter title:

**Problem solving with c++**

Enter author:

**Savitch, walter**

Enter date:

**2006**

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 2**

The books entered so far, sorted alphabetically by author are:

Savitch, walter. Problem Solving with C++. 2006.

Sturgeon, Theodore, More Than Human. 1953.

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 1**

Enter title:

**At Home in the Universe**

Enter author:

**Kauffman**

Enter date:

**1996**

Select from the following choices:

1. Add new book
2. Print listing sorted by author
3. Quit

**Choice : 2**

The books entered so far, sorted alphatetically by artist are:

Kauffman, At Home in the Universe, 1996.

Savitch, walter. Problem Solving with C++. 2006.

Sturgeon, Theodore, More Than Human. 1953.

**Book.h**

#ifndef BOOK\_H

#define BOOK\_H

#include <string>

using namespace std;

class Book

{

public:

Book();

Book(string new\_author, string new\_title, string new\_date);

void setData(string new\_author, string new\_title, string new\_date);

string getAuthor() const;

string getTitle() const;

string getDate() const;

friend bool operator< (const Book &book1, const Book &book2);

private:

string author, title, date;

};

#endif

**Book.cpp**

Add the definitions of data members of class Book. The definition of friend function should be (in Book.cpp):

bool operator< (const Book &book1, const Book &book2)

{

return (book1.author < book2.author);

}

**TestProgram.cpp**

Don’t forget to add header files, vector and algorithm along with other required header files.

using namespace std;

void AddNewBook(vector<Book> \*bookdata);

void PrintBooks(vector<Book> &bookdata);

void SortBooks(vector<Book> &bookdata);

void PrintMenu();

int main()

{

vector<Book> bookdata;

//Add your logic to test your class, using functions mentioned above

return 0;

}

**Program # 2**

Write a program to implement English Dictionary that contains data members as word, meaning, synonym, antonym and example\_sentence. Use an appropriate STL container and apply the relevant algorithms to maintain the sense of Dictionary.

**Part B**

**Program #3**

A frequency table lists words and the number of times each word appears in a text file.  
Write a program that creates a frequency table for a file whose name is entered by the  
user. You can use a map of string-int pairs. You may want to use the C library function  
ispunct() (in header file CTYPE.H) to check for punctuation so you can strip it off the  
end of a word, using the string member function substr(). Also, the tolower() function  
may prove handy for uncapitalizing words.

**Program #4**

Create a single linked list that contains the data (age) of your friends. Perform basic operations including insertion, deletion, searching and display. The insertion operation should only allow a friend’s data to be inserted in sorted order only.

**Part C**

**Program #5**

Create a student hash table that contains information, studentID (int), name (string), marks\_oop345 (float). The size of hash table is equal to the number of students in the class. Use linear probing in case of collisions. Perform insertion, deletion, display and search operations.