CS 201, Spring 2019
Homework Assignment 1

Due: 23:55, April 4 (Thursday), 2019

In this homework, you will implement a bookstore system to store multiple books. Each book has its title, its price, and a list of the years of its editions. The books in the bookstore system and the list of their edition years have to be implemented using **dynamically allocated arrays**. The homework has two parts whose requirements are explained below.

PART A (30 points):

To take the final exam, you must submit this part and receive at least half of its points.

This part includes the implementation of the `Book` class, which is used to implement a single book. This class stores the title and price of the book as well as a list of the years of its editions (please see the data members of the given `Book` class definition). The list of the edition years has to be stored in a dynamically allocated array of integers. You will use this class definition and implementation in Part B of your homework.

Note that this class definition has to be stored in a header file named `Book.h`. This header file is provided to you and you are not allowed to change the contents of this file.

```
#ifndef __BOOK_H
#define __BOOK_H

#include <string>

class Book{
public:
   Book( const string bookTitle = "", const double bookPrice = 0 );
   Book( const Book& bookToCopy );
   ~Book();
   Book& operator=( const Book& right );
   void addEdition( const int newEditionYear );

private:
   string title;         // Title of the book
   double price;         // Price of the book
   int editionNo;        // Number of the editions of the book
   int* editionYears;    // A list of the years of the book's editions
                         // Note that each book can have zero or more editions

// BookStore class, which you will implement in Part B, is declared as friend
friend class BookStore;

// Overloaded global functions for >> and << are declared as friend
friend istream& operator>>( istream& in, Book& b );
friend ostream& operator<<( ostream& out, const Book& b );
};

#endif
```

You need to implement each of the given functions in a source file named `Book.cpp`. The details of these functions are given below:

- The constructor initializes the title and the price of the book with the provided arguments. If none provided, the default values of "" and 0 should be used, as given in the class definition. This constructor needs to create an empty array for the years of the book's editions.
- The destructor needs to be implemented to avoid memory leaks.
- The copy constructor and the assignment operator need to perform a deep copy operation.
- The `addEdition` function adds an edition year to the book. Edition years do not need to be unique.
- The `operator<<` function is a global function that outputs the information of the book object given on the right hand side of `<<`. Your output should exactly have the format given below. Otherwise, you will lose points.

  The first line gives the format when the number of editions is zero. The second line gives the format when there is one edition. The third line gives the format when there are more than one edition. You can find examples in the `OutputPartA.txt` file, which includes the output of an example code given in the `mainPartA.cpp` file.

  title-of-the-book, *price-of-the-book* TL (none)

  title-of-the-book, *price-of-the-book* TL (*1st-year*)

  title-of-the-book, *price-of-the-book* TL (*1st-year*, *2nd-year*, ..., *last-year*)
- The `operator>>` function is a global function that inputs the edition years of the book object given on the right hand side of `>>`. This function needs to input all edition years typed in a single line terminated by a new line character. The edition years are separated with whitespace characters. Here you may assume that the user only inputs digits and whitespace characters. Note that if this line just includes whitespace characters (without any digits) or if it is empty, the given book object will have an empty list of edition years. Some input examples are provided at the end of the `mainPartA.cpp` file.

  The `operator>>` function overwrites the edition years of the book object given on the right hand side of `>>`. That is, before getting the new edition year(s) from the input stream, you have to delete the existing edition years of the given object to avoid memory leaks.

  This function requires getting all of the characters in a single line into a string and tokenizing the string into individual years (which are of the integer type). As a part of this homework, every student will study and learn (by his/her own) how to use the string functions and/or how to manipulate strings in order to extract the required tokens from a given string.

PART B (70 points):

This part includes the implementation of the `BookStore` class. This class stores the number of the books in the bookstore system and the book objects (please see the data members of the given `BookStore` class definition). The book objects have to be stored in a dynamically allocated array.

Note that this class definition has to be stored in a header file named `BookStore.h`. This header file is provided to you and you are <u>not allowed to change</u> the contents of this file.

```
#ifndef __BOOKSTORE_H
#define __BOOKSTORE_H

#include "Book.h"

class BookStore{
public:
   BookStore();
   BookStore( const BookStore& bsToCopy );
   ~BookStore();
   BookStore& operator=( const BookStore& right );
   Book& operator[]( const string title );

   void addBook( const string bookTitle, const double bookPrice );
   void removeBook( const string bookTitle );

private:
   Book* books;         // A dynamically created array of book objects
   int bookNo;          // Number of the books in the bookstore system

// Overloaded global function for << is declared as friend
friend ostream& operator<<( ostream& out, const BookStore& b );
};

#endif
```

You need to implement each of the given functions in a source file named `BookStore.cpp`. The details of these functions are given below:

- The constructor creates an empty array of books.
- The destructor needs to be implemented to avoid memory leaks.
- The copy constructor and the assignment operator need to perform a deep copy operation.
- The overloaded subscript operator takes a title as its input parameter and returns the book object with this title. It returns the book object by reference such that the returned object can be used as an r-value as well as an l-value. This overloaded operator throws an exception if its argument is invalid. In other words, it throws an exception if the bookstore system does not have a book with the given title.
- The `addBook` function adds a new book to the bookstore system. The title and the price of the book are specified as input parameters. This function does not add any edition years; that is, the edition years of the book should be initialized with an empty array. The book titles are unique (case sensitive). If the system already has a book with the specified title, this function does not add this book to the system and throws an exception.
- The `removeBook` function removes an existing book from the bookstore system. The title of the book is specified as an input parameter. It also clears the edition years of the specified book. If the system does not have a book with the specified title, this function throws an exception.
- The `operator<<` function is a global function that outputs the information of all of the books in the bookstore object given on the right hand side of `<<`. Its output should contain an output line for each book; the format of this line has been explained in Part A. You can also find examples in the `OutputPartB.txt` file, which includes the output of an example code given in the `mainPartB.cpp` file.

WHAT TO SUBMIT?

You have to submit **only two source code files**. These are `Book.cpp`, which includes the implementation of the `Book` class, and `BookStore.cpp`, which includes the implementation of the `BookStore` class. (If you do not implement Part B, do not submit the `BookStore.cpp` file.)

Upload your homework as a zip file (secX-firstname-lastname.zip where X is your section number) using the upload link on Moodle (All Sections) before the deadline. This upload page will be available between March 18 and April 7, 23:55.

The standard rules about late homework submissions apply. Late submissions need to be done by emailing your zip file to your TA Furkan Hüseyin.

Besides these two source code files, do not submit anything (no header files or no driver files). You will lose 5 points for each additional file that you will submit.

NOTES ABOUT IMPLEMENTATION

1. You must use dynamically allocated arrays. You will receive no points if you use fixed-sized arrays, linked-lists or any other data structures such as `vector/array` from the standard library.
2. You cannot use any global variables in your implementation.
3. You cannot define additional member functions or global functions when implementing the classes. You cannot change the header files. Do not forget that you cannot submit your header files and you have to use the ones that we have provided to you.
4. Your code must not have any memory leaks. You will lose points if your code has memory leaks even though it produces correct outputs. To detect memory leaks, you may want to use Valgrind which is available at http://valgrind.org.
5. The `mainPartA.cpp` and `mainPartB.cpp` driver files are just given as examples. They do not test all aspects of your implementation. Thus, we will use other driver files (other test cases) to grade your assignment. Thus, you have to test your implementation thoroughly by writing your own driver files. Indeed, knowing how to test a program is crucial in designing and implementing any software. Thus, as a part of this homework, every student needs to experience (and hopefully to learn) how to test a simple program.
6. You are free to code your program on Linux or Windows platforms. However, we will test your program on "`dijkstra.ug.bcc.bilkent.edu.tr`" and we will expect your program to compile and run on the `dijkstra` machine. If we cannot get your program to properly work on the `dijkstra` machine, you will end up losing a considerable number of points (at least 10 points for Part A and 20 points for Part B). Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra.ug.bcc.bilkent.edu.tr`" before submitting your assignment.
7. This assignment will be graded by your TA **Furkan Hüseyin** (**`furkan.huseyin at bilkent edu tr`**). Thus, you may ask your homework related questions directly to him.

VERY IMPORTANT: We expect all of you to comply with academic integrity. The honor code statement, which has been sent you by email, was prepared to clarify our expectations about the academic integrity principles. Please study the part of this statement related with "individual assignments" very carefully.

If you submit this homework assignment, we will assume that you all read this honor code statement and comprehensively understood its details. Thus, please make sure that you understood it. If you have any questions (any confusions) about the honor code statement, consult with the course instructors.

We will check your codes for plagiarism.