CLASSES AND PRIVACY

Workshop 3 (10 marks – 3.75% of your final grade)

In this workshop, you are to code a class with private data members and public member functions. The class will represent a book and can be used to manage a collection of books that a person has in her possession.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to

- define a class type;
- privatize data within the class type;
- instantiate an object of class type;
- access data within an object of class type through public member functions;
- use standard library facilities to format data inserted into the output stream;
- truncate a C-style null-terminated string to fit in memory
- describe what you have learned in completing this workshop.

SUBMISSION POLICY

The *in-lab* section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period.

If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. You must be present at the lab in order to get credit for the *in-lab* portion.

If you do not attend the lab, you can submit the *in-lab* section along with your *at-home* section (see penalties below). The *at-home* portion of the lab is due on the day that is four days after your scheduled *in-lab* workshop (@23:59) (even if that day is a holiday).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to back up your work regularly.

LATE SUBMISSION PENALTIES:

- *In-lab* portion submitted late, with *at-home* portion: **0** for *in-lab*. Maximum of **7**/10 for the entire workshop.
- If any of *in-lab*, *at-home* or *reflection* portions is missing, the mark for the workshop will be **0**/10.

In-Lab (30%)

Design and code a module named `Book` that this workshop's application can use. The application is listed below. Store your class definition in a header file named `Book.h` and your function definitions in an implementation file named `Book.cpp`.

Include a **compilation safeguard** in the header file. Enclose all declarations and definitions within the `sict` namespace.

In the `Book.h` header file, predefine the following constants as integers:

`max_title_size` with the value 32. This represents the maximum number of characters for the title of the book (not including the null byte `'\0'`)

`max_name_size` with the value 16. This represents the maximum number of characters for the name of the book author (not including the null byte `'\0'`)

Each book in the world is uniquely identified by an ISBN-13 number. This is a number that has **exactly** 13 digits (the previous standard, now deprecated, used only 10 digits for ISBN).

Also, in the `Book.h` header file predefine the following constants (read the information at https://docs.microsoft.com/en-us/cpp/cpp/data-type-ranges to select an appropriate **C++ standard** type, capable of storing numbers with 13 digits):

`min_isbn_value` with the value 1,000,000,000,000, representing the smallest ISBN that can be accepted

`max_isbn_value` with the value 9,999,999,999,999, representing the largest ISBN that can be accepted by the application.

Your `Book` module validates any ISBN that it receives. For this *in-lab* submission, an ISBN is valid if it is a 13-digit number between the predefined limits.

Define the `Book` class with private members that hold the following data:

- The family name on the author
- The given name on the author
- The ISBN that identifies the book

Declare the following public member functions in your class definition:

`void set(...)`: This function receives 4 parameters representing the *author given name*, the *author family name*, the *title* of the book and the *ISBN* that identifies the book.

This function checks if ISBN is a valid one (for the purpose of the *in-lab* portion, a valid ISBN is any integer that has **exactly 13 digits**). If so, this function stores the received parameters in the current object. If the ISBN is not valid, this function stores values that identify an empty state. It is up to you to select the data values that identify the state as empty.

In designing this function make sure that it does not overflow the space allocated for the title, family and given names. Store only as many characters as there is space available. You may use the `strncpy(...)` function in the `<cstring>` library for this purpose.

`bool isEmpty() const`: This function returns `true` if the object is in a safe empty state, `false` otherwise.

`void display() const`: If the object is not empty, this function inserts into the standard output stream the object's data in the following format:

```
Author: FAMILY_NAME, GIVEN_NAME<ENDL>
Title: TITLE<ENDL>
ISBN-13: ISBN<ENDL>
```

If the object is in an empty state, this function should print:

```
The book object is empty!<ENDL>
```

Below the source code is the expected output from your program. The **green color** identifies what is generated by your program. The output of your program should match **exactly** the sample output shown below.

In-Lab Main Module

```
#include <iostream>
#include "Book.h"

using namespace std;
using namespace sict;

int main()
{
	cout << "Book Management App" << endl;
	cout << "===================" << endl;
	Book aBook;

	cout << "Testing that validation and display are correct:" << endl;
	cout << "-------------------------------------------------" << endl;
	aBook.set("Frank", "Herbert", "Dune", 91780441172719LL);
	aBook.display();
	cout << "The Book::isEmpty() should return true --> "
	     << (aBook.isEmpty() ? "correct" : "incorrect") << endl;

	aBook.set("Frank", "Herbert", "Dune", 980441172719LL);
	aBook.display();
	cout << "The Book::isEmpty() should return true --> "
	     << (aBook.isEmpty() ? "correct" : "incorrect") << endl;

	aBook.set("Frank", "Herbert", "Dune", 9780441172719LL);
	aBook.display();
	cout << "The Book::isEmpty() should return false --> "
	     << (aBook.isEmpty() ? "incorrect" : "correct") << endl;

	return 0;
}
```

IN-LAB EXPECTED OUTPUT

```
Book Management App
===================
Testing that validation and display are correct:
------------------------------------------------
The book object is empty!
The Book::isEmpty() should return true --> correct
The book object is empty!
The Book::isEmpty() should return true --> correct
Author: Herbert, Frank
Title: Dune
ISBN-13: 9780441172719
The Book::isEmpty() should return false --> correct
```

IN-LAB SUBMISSION

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload `Book.h`, `Book.cpp` and `w3_in_lab.cpp` to your matrix account. Compile and run your code and make sure everything works properly.

Then, run the following command from your account (use your professor's Seneca userid to replace `profname.proflastname`, and your section ID to replace XXX, i.e., SAA, SBB, etc.):

```
~profname.proflastname/submit 244XXX_w3_lab<ENTER>
```

and follow the instructions.

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If your professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

AT-HOME (30%)

The *at-home* section of this workshop upgrades your `Book` module to store more information about a book, improve the validation of the ISBN and display the information about a book in an easier to read format.

For the at-home portion of the workshop, you are going to improve the procedure that validates an ISBN.

A valid ISBN must have ***exactly*** *13 digits. The right most digit is a check digit that validates the other digits. For a number to be a valid ISBN, the weighted sum of all 13 digits must be divisible by 10.*

To obtain the weighted sum, remove the check digit (the right-most digit). From the remaining 12 digits, select those that are on odd positions (position 1, 3, 5, 7, 9, 11) and add them together. Then select the remaining digits, add them together and multiply the result with 3. Add the two sums to obtain a total. If the difference between the next highest number ending in zero and this total is check digit, then the whole number is a valid ISBN; otherwise, the whole number is not a valid ISBN.

```
ISBN  9780441172719
                  |
                  9 – this is the check digit
      9 8 4 1 7 7   – these are the digits on odd positions
       7 0 4 1 2 1  – these are the digits on even positions

Calculate the first sum: 9 + 8 + 4 + 1 + 7 + 7 = 36
Calculate the second sum and multiply it by three: (7 + 0 + 4 + 1 + 2 + 1) × 3 = 45
Add together the two values: 36 + 45 = 81
Next highest integer multiple of 10  = 90
Difference                           =  9
 Matches the check digit, therefore this number is a valid ISBN
```

Copy the original module to your *at-home* directory.

Add the following attributes to your `Book` class:

- An integer that represents the year when the book was published
- A floating point number in double precision representing the price of the book

Add a new public member function:

`void set(int year, double price)`: if the object is not empty, this function stores the parameters into the attributes. If the object is empty, this function does nothing.

Update the implementation of the existing members:

`void set(...)`: change the existing function with 4 parameters to use the new logic to validate the ISBN (as described above).

`void display(...) const`: add a Boolean parameter to this function with a **default value** as `false`. This parameter signals if the content of the object should be printed on a single line or on multiple lines.

If the object is **empty** and the parameter is `false`, this function should print:

```
The book object is empty!<ENDL>
```

If the object is **not empty** and the parameter is `false`, this function inserts into the standard output stream the object's data in the following format:

```
Author: FAMILY_NAME, GIVEN_NAME<ENDL>
Title: TITLE<ENDL>
ISBN-13: ISBN<ENDL>
Publication Year: YEAR<ENDL>
Price: PRICE<ENDL>
```

If the object is **empty** and the parameter is `true`, this function should print the text "`The book object is empty!`" in a *left* aligned field of width 92, like this:

```
|The book object is empty!                                                                  |<ENDL>
```

If the object is **not empty** and the parameter is `true`, this function inserts into the standard output stream the object's data on a single line:

- Author's family name: a *right* aligned field of width `max_name_size`
- Author's given name: a *right* aligned field of width `max_name_size`
- Title: a *left* aligned field of width `max_title_size`
- ISBN: a *right* aligned field of width 13
- Year: a *right* aligned field of width 4
- Price: a *right* aligned field of width 6, and precision 2

The fields are separated by the pipe symbol (`'|'`), like this:

```
|     FAMILY_NAME|      GIVEN_NAME|TITLE                           |      ISBN-13|YEAR| PRICE|<ENDL>
```

Below the source code is the expected output from your program. The **green color** identifies what is generated by your program. The output of your program should match **exactly** the sample output shown below.

AT-HOME MAIN MODULE

```
#include <iostream>
#include "Book.h"

using namespace std;
using namespace sict;

int main()
{
   cout << "Book Management App" << endl;
   cout << "===================" << endl << endl;

   cout << "Checking the constants:" << endl;
   cout << "-----------------------" << endl;
   cout << "max_title_size = " << sict::max_title_size << endl;
   cout << "max_name_size = "  << sict::max_name_size << endl;
   cout << "min_isbn_value = " << sict::min_isbn_value << endl;
   cout << "max_isbn_value = " << sict::max_isbn_value << endl << endl;

   sict::Book books[10];
   books[0].set("Frank",  "Herbert", "Dune", 91780441172719LL); // invalid
   books[0].set(1990, 9.73);
   books[1].set("Frank",  "Herbert", "Dune", 980441172719LL);   // invalid
   books[1].set(1990, 9.73);
   books[2].set("Frank",  "Herbert", "Dune", 9780441172718LL);  // invalid
   books[2].set(1990, 9.73);
   books[3].set("Frank",  "Herbert", "Dune", 9780441172709LL);  // invalid
   books[3].set(1990, 9.73);
   books[4].set("Frank",  "Herbert", "Dune", 9780441172619LL);  // invalid
   books[4].set(1990, 9.73);
   books[5].set("Frank",  "Herbert", "Dune", 9780441172719LL);  // valid
   books[5].set(1990, 9.73);
   books[6].set("George", "Orwell",  "1984", 9780451524935LL);  // valid
   books[6].set(1950, 6);
   books[7].set("Jane",   "Austen",  "Pride and Prejudice", 9780199535569LL);  // valid
   books[7].set(2008, 6.1);
   books[8].set("J.R.R.", "Tolkien", "The Lord of the Rings", 9780544003415LL);  // valid
   books[8].set(2012, 12.8);
   books[9].set("Harper", "Lee",     "To Kill a Mockingbird", 9780446310789LL);  // valid
   books[9].set(1988, 10.99);

   cout << "Displaying the library as a list:" << endl;
   cout << "---------------------------------" << endl;
   for (int i = 0; i < 10; ++i)
   {
          if (books[i].isEmpty() == false)
                 cout << endl;
          books[i].display();
   }
```

```
	cout << endl << endl;

	cout << "Displaying the library as a table:" << endl;
	cout << "-----------------------------------------------"
	     << "-----------------------------------------------" << endl;
	for (int i = 0; i < 10; ++i)
	{
		books[i].display(true);
	}
	cout << "-----------------------------------------------"
	     << "-----------------------------------------------" << endl;

	return 0;
}
```

At-Home Expected Output

```
Book Management App
===================

Checking the constants:
-----------------------
max_title_size = 32
max_name_size = 16
min_isbn_value = 1000000000000
max_isbn_value = 9999999999999

Displaying the library as a list:
---------------------------------
The book object is empty!
The book object is empty!
The book object is empty!
The book object is empty!
The book object is empty!

Author: Herbert, Frank
Title: Dune
ISBN-13: 9780441172719
Publication Year: 1990
Price: 9.73

Author: Orwell, George
Title: 1984
ISBN-13: 9780451524935
Publication Year: 1950
Price: 6

Author: Austen, Jane
Title: Pride and Prejudice
ISBN-13: 9780199535569
Publication Year: 2008
Price: 6.1

Author: Tolkien, J.R.R.
Title: The Lord of the Rings
```

```
ISBN-13: 9780544003415
Publication Year: 2012
Price: 12.8

Author: Lee, Harper
Title: To Kill a Mockingbird
ISBN-13: 9780446310789
Publication Year: 1988
Price: 10.99


Displaying the library as a table:
----------------------------------------------------------------------------------------------
|The book object is empty!                                                                   |
|The book object is empty!                                                                   |
|The book object is empty!                                                                   |
|The book object is empty!                                                                   |
|The book object is empty!                                                                   |
|         Herbert|           Frank|Dune                            |9780441172719|1990|  9.73|
|          Orwell|          George|1984                            |9780451524935|1950|  6.00|
|          Austen|            Jane|Pride and Prejudice             |9780199535569|2008|  6.10|
|         Tolkien|          J.R.R.|The Lord of the Rings           |9780544003415|2012| 12.80|
|             Lee|          Harper|To Kill a Mockingbird           |9780446310789|1988| 10.99|
----------------------------------------------------------------------------------------------
```

Reflection (40%)

Study your final solution, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time.**

Create a file named `reflect.txt` that contains your **detailed description of the topics that you have learned** in completing this workshop and mention any issues that caused you difficulty. Include in your explanation—**but do not limit it to**—the following points:

1) What type have you selected for ISBN? Explain your reasoning.
2) For the *at-home* portion you had to change the logic that validates an ISBN. How would you design your class in such a way that if a new update to the validation logic is necessary, you don't have to change anything in the function `Book::set(...)`?

Quiz Reflection

Add a section to `reflect.txt` called Quiz X Reflection. Replace the X with the number of the last quiz that you received and list the numbers of all questions that you answered incorrectly.

Then for each incorrectly answered question write your mistake and the correct answer to that question. If you have missed the last quiz, then write all the questions and their answers.

At-Home Submission

To submit the *at-home* section, demonstrate execution of your program with the exact output as in the example above.

Upload `reflect.txt`, `Book.h`, `Book.cpp` and `w3_at_home.cpp` to your matrix account. Compile and run your code and make sure everything works properly.

To submit, run the following command from your account (use your professor's Seneca userid to replace `profname.proflastname`, and your section ID to replace `XXX`, i.e., SAA, SBB, etc.):

```
~profname.proflastname/submit 244XXX_w3_home<ENTER>
```

and follow the instructions.

Important: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.