CS2400 Book Class (Part 1) Due: Thursday November 19, 2020
55 Points

Objectives:

Design classes, search strings, tokenizing strings, string member functions, operator overloading, and separate compilation.

Important:

Do not change the function names or their prototypes. Also, use the same file names given in the repository on GitHub (book.h, book.cc).

You may use any function or library discussed in class or in the chapters we covered from your textbook. Do not use any other libraries or functions.

Design a class to keep track of a book. Each book has a *title*, *year*, and an *author*. The year must always be zero or above. The title and the author must never be empty. The member functions should always check for valid data.

Your class must include the following constructors and functions (**function names and prototype must match exactly**):

- `Book()`
  - A constructor that initializes a book to the default values: `("***", 0, "***")`.
- `Book(string newTitle, int newYear, string newAuthor);`
  - A constructor that initializes a book's title, year, and author to the specified parameters.
- `Book(string allData)`
  - A constructor that splits the string specified (`allData`) into the three book properties. The string is in the following format:
    - `title|year|author`
    - Example:
      `"The Hitchhiker's Guide to the Galaxy|1979|Douglas Adams"`
- Setters (mutators) for all three member variables (e.g. `setTitle`)
- Getters (accessors) for all three member variables (e.g. `getTitle`)
- A member function called `toString` that returns all the book data as a single string in the format: `"title|year|author"`. Do not add any extra spaces around "|".
  For example:
  `"The Hitchhiker's Guide to the Galaxy|1979|Douglas Adams"`
- `bool matchTitle(string targetTitle)`
  - Returns `true` if `targetTitle` is part of the book title
  - Using the example above it should return `true` if `targetTitle` is `"galaxy"`, `"GUIDE tO"`,`"the hit"`, etc.
- `bool matchAuthor(string targetAuthor)`
  - Returns `true` if `targetAuthor` is part of the name of the lead author
  - Using the example above it should return true if `targetTitle` is `"Doug"`, `"LAS"`, `"aDaMs"`, etc.

- `bool matchYear(string targetYear)`
  - Returns `true` if `targetYear` matches any part of the year.
  - Using the example above it should return `true` if `targetYear` is `"79"`, `"19"`, or `"1979"` etc.
- `bool match(string target);`
  - Returns `true` if target can be found any where in the book member variables
    - Using the example above it should return `true` if target `is "97", "douglas", "hitch",` etc.

Write a main program to test your code or use the unit tests provided on GitHub. The unit tests provided test all the functions above based on the specifications given. If you get the green checkmark ✓, it means you implemented all the class functions correctly. If you get the red 'X', please check to see which one of your functions failed.

Project Files:
Divide your project into three files:

- `book.h`
  - Contains the class definition
- `book.cc`
  - Contains the class implementation (all the functions)
- `book-main.cc`
  - Main program to test your class

Compiling your project:

1. `g++ -Wall -c book.cc`
   - This creates the object file `book.o`
2. `g++ -Wall -c book-main.cc`
   - This creates the object file `book-main.o`
3. `g++ book.o book-main.o`
   - This creates the executable `a.out`

or

```
g++ -Wall -std=c++11 book.cc book_main.cc
```

If you have the make utility installed on your system, there is a `Makefile` provided to allow you to compile your program using the command:

```
make
```

If you are using C++11 add the option `-std=c++11` to your compile commands.

Hints:

- Start early. You may start by putting everything in one file and separate them later
- Implement the first two constructors
- Implement the toString function so you can output an object
- Implement the getters, and setters functions first
- Implement the *match* functions one at a time. Test every function immediately after you write it.
- Review the *string* functions. They will be useful in this project.
- Write a function that converts a string to all upper case.
- The function `stoi` allows you to convert a string object to an integer. It requires compiling your program with `-std=c++11` option.
  `int x = stoi("123");` will return the integer 123
- The function `to_string` converts a number to string.
  `string s = to_string(123);` will set s to the string `"123"`

- For the third constructor, use the `find` and the `substr` string member functions.

Grading:

Programs that contain syntax errors will earn zero points.
Programs that use global variables, other than constants, will earn zero points.
Programs that do not use separate files will earn zero points.

- (3 points) default constructor
- (3 points) second constructor
- (6 points) third constructor
- (6 points) getters and setters
- (4 points) toString function
- (28 points) match functions (7 points each)
- (5 points) Programming Style & documentation.
  - All files must have your name, date, and a description of their content
  - All functions must be documented in the header file using the style outlined in the coding style below

Follow the coding style outline on GitHub:
https://github.com/nasseef/cs2400/blob/master/docs/coding-style.md

Submission:

Submit a link to your repository before the deadline.