Larger Assignment, Part 1.

"A library implies an act of faith."
– Victor Hugo

To begin with our library building project, we are going to make two classes this week: Author and Book.

The tester sample code is online. Do not modify it, simply create the two classes and the associated methods it expects to run.

An **Author** is a writer of books.
They have a first and last name and birth/death dates.

An **Author** has two constructors:

`Author()`
blank constructor, creates a new Author with no name, birth, or death dates set

`Author( last name, first name )`
creates a new Author with a name, but birth and death dates not set

An **Author** has at least the following methods:
(You can make more if you feel you need to, for helping your other methods, but I will be calling and testing these.)

`int getBirth()`
returns an int representing the author's year of birth

`int getDeath()`
returns an int representing the author's year of death

`void setDates( birth )`
sets the author's year of birth, must be a valid year

`void setDates( birth, death )`
sets both the author's year of birth and death, must be valid years, and the year of death must be after the year of birth

`int compareTo( Author )`

compares two authors. This is similar to Java's String compareTo() method. If two authors have the same first name and last names it should return 0, otherwise it should return not 0.

Bonus: Sometimes authors do not always write out their entire first name. For example Julian Barnes and J Barnes could be considered equivalent. However, Julian Barnes and John Barnes should not be. Make your compareTo more flexible by returning 0 if a first initial and last name match.

`String toString()`

Should return a string the author, of the form: Barnes, Julian

`String printAuthorInfo()`

Should **return** (not print) a string the author, of the form:

```
Heaney, Seamus
```

If the year of birth (but not death) is set it should return:

```
Heaney, Seamus (b. 1939)
```

If both the years of birth and death have been set, it should return:

```
Heaney, Seamus (1939-2013)
```

Note a valid year, is one after the year -2000 (I am letting that represent BC) and before the year 2018 (some books do announce their publication in advance, though rarely more than five years out, most people have not forecast their births or deaths more than five years out).

A Book is our main object.
Books in our system have a date (year) of being written, a title, an ISBN, and a single Author (Author object) who wrote them.

A **Book** has three constructors:

`Book()`

blank constructor, creates a new Book with no title, year of publication, Author, or ISBN set

`Book( title )`

creates a new Book with a title, but no year of publication, Author, or ISBN set

`Book( title, Author )`

creates a new Book with a title and an Author (as an Author object) but no year of publication or ISBN set

A **Book** has at least the following methods:
(Again, you can make more if you feel you need to, for helping your other methods, but I will be calling and testing these.)

`void setTitle( title )`
sets the book's title must be a not-empty title
`String getTitle()`
returns a String representing the book's title
`void setAuthor( Author )`
sets the Author object
`Author getAuthor()`
returns the Author object who wrote the book
`void setYear( year )`
sets the year the book was written/published, must be a valid year
`int getYear()`
returns the year the book was written
`void setISBN ( ISBN )`
sets the ISBN for the book, this must be of length 13 or length 10 (we will be omitting all dashes)
`String getISBN()`
returns a String representing the ISBN of the book
`boolean sameAuthor( Book )`
returns a boolean which is true if this book has the same Author as the book that was passed in in the arguments, or otherwise false (hint: you wrote a method to compare authors.)
`int compareTo ( Book )`
compares two Book objects, for now this can be done ONLY by comparing ISBNs.
`String toString()`
Should return (not print!) a String representation of this Book.
If only a title has been set, return a string of the form:

```
The Arcades Project.
```

If a title and author are set:

```
The Arcades Project. Benjamin, Walter.
```

If a title and author and year have been set:

```
The Arcades Project (2002). Benjamin, Walter.
```

Note the punctuation above, it must be exact to pass the tests.

Note, when trying to run a setter method with an unallowable variable, please keep the previous value. Returning error messages is a good idea, however the text of them is up to you.

Finally, there are a series of constants at the top of the sample file. Please use these variables for properties of books and authors that have not yet been set.

Final output if achieving all points:

```
Attempting constructors:
  - Author constructors seem functional:  3/3
  - Book constructors seem functional:    3/3

Attempting simple getters/setters:
  - getTitle/setTitle:                    2/2
  - getAuthor/setAuthor:                  2/2
  - getYear/setYear:                      2/2
  - getISBN/setISBN:                      2/2
  - author getBirth/getDeath/setDates:  4/4

Attempting un-set values register correctly:
  - un-set values:  5/5

Attempting bad-value setters:
  - year too far in the future rejected:  2/2
  - empty title rejected:                 2/2
  - ISBN of incorrect length rejected:    2/2
  - year of birth in super past rejected: 2/2
  - year of death before birth rejected:  2/2

Comparisons:
  - author comparison tests:          3/3
  - *bonus* author comparison tests:  5/5
  - sameAuthor (across two books):    2/2
  - comparing two books:              2/2

Printing authors and books:
  - printing an author's name:          2/2
  - printing an author's name + info:  3/3
  - printing a book's name + info:     5/5

Overall Score: 55/50
   (can be up to 55 with bonus)
```

Larger Assignment, Part 2.

You already did the hard work!
You structured two classes, you built them and they work!

Now we are going to build a bigger class called Library that uses your Book and Author (and their important methods).

The code compiles, right from the start!, but you get 0 points.

To get all the points, fill in the 12 methods listed below!

`int totalCopies()`
this method should return an integer that is the number of total copies of all books that exist in the library (0 to n)

`int checkedOut()`
this method should return an integer that is the number of total checked out books at the current time (0 to n)

`String printStatus()`
this method should print out just the "status" of the library, in the format:

```
Total unique books: 26
Total number of copies: 40
Total checked out: 0
```

`void addMultipleBooks( Book [] b )`
this method should add an entire array of books into the library Note their may be other books in the library already so you cannot simply overwrite the entire books array, but rather must add these *at* the end of the current library. Make sure to include one copy of each book. [For simplicity no books added through this method will ever be duplicates of each other or previous books, promise.]

`void addBook( Book b )`
this method adds a single book to the library.
If the book is already present, it should add another copy of a pre-existing book. If the book is new then it should be added to the end of an array.

`String checkOut ( Book b )`

this method checks out a book from the library IF the book exists AND there are remaining copies which haven't been checked out. This method returns a string denoting success, either:

```
"Book not found."
"All out of copies."
"Checked out!"
```

`String checkIn ( Book b )`

this method checks in a book to the library IF the book exists in the library collection AND there are copies which have already been checked out. This method returns a string denoting success, either:

```
"Book not found."
"All of our copies are already checked in."
"Checked in!"
```

`String printLibrary()`

this method prints out the entire library collection, including status, where book numbering starts at 0, and the numbers after the colon denote current copies, and total copies:

```
0. Ulysses. Joyce, James. : 1/1
1. The Great Gatsby. Fitzgerald, F. Scott. : 1/2
2. Lolita. Nobokov, Vladimir. : 1/1
3. Brave New World. Huxley, Aldous. : 1/1
4. The Sound and the Fury. Faulkner, William. : 2/2
5. Catch-22. Heller, Joseph. : 1/1
6. Darkness at Noon. Koestler, Arthur. : 1/1

Total unique books: 7
Total number of copies: 9
Total checked out: 1
```

`int numBooksByAuthor( Author a )`

this method returns an int that specifies how many unique books exist for a given author (0 to n)

`String booksByAuthor( Author a )`

this method returns a String that lists the unique books which exist for a given author, in standard book format:

```
Pitch Dark. Adler, Renata.
Speedboat. Adler, Renata.
```

for simplicity, you may include a line break “\n” after each title (including the last one). If no books are found this method should return:

```
No books by Adler, Renata.
```

`String booksByTitle( String find )`

this method returns a String that lists the unique books which exist with a given String anywhere in the book’s titles.
For example a search for “of” would return:

```
A Portrait of the Artist as a Young Man. Joyce, James.
The Grapes of Wrath. Steinbeck, John.
The Way of All Flesh. Butler, Samuel.
The Wings of the Dove. James, Henry.
```

for simplicity, you may include a line break “\n” after each title (including the last one).
alternatively, a search for “cloud atlas” would return:

```
No books with "cloud atlas" in the title.
```

`String deleteBook( Book b )`

this method deletes a book entirely from the list of books. It must update all arrays to not leave a hole in the lists of books. Bonus! Good luck!

Final output if achieving all points:

```
Let's calculate some facts about the library...
 - totalCopies exists: 2/2
 - checkedOut exists: 2/2
 - printStatus exists: 3/3

Let's add a set of books, and add more books...
 - setBooks seems to work: 3/3
 - addBooks seems to work for duplicates: 2/2
 - addBooks seems to work for new books: 2/2
 - addMultipleBooks works with non-empty libraries: 2/2
 - printStatus works after setBooks/addBooks: 2/2

Let's try checking out books...
 - checkOut works when a book can be checked out: 2/2
 - checkOut works when a book is out of copies: 2/2
 - checkOut works when a book is not in the library: 2/2
 - printStatus works after checking out books: 3/3

Let's try returning books...
 - checkIn works when a book is not in the library: 2/2
 - checkin works when a book is not checked out: 2/2
 - checkIn works when a book is checked back in: 2/2
 - printStatus works after checking out books: 3/3

Let's try to print the whole library...
 - printLibrary works: 3/3

Let's ask the library some questions...
 - numBooksByAuthor works: 3/3
 - booksByAuthor works: 4/4
 - booksByTitle works: 4/4

Bonus! Let's remove a book from the library forever...
 - book deletion: 5/5

Total score: 55/50
   (can be up to 55 with bonus)
```