CS 1410: ISBN Index

If you're familiar with books, then you're probably familiar with an ISBN. ISBN is an acronym for International Standard Book Number. In many countries, an ISBN agency exists to assign a unique ISBN to each book that is published in that country. By doing so, the vast majority of books that get published internationally can be identified by a single, unique ISBN, and this makes it easy for anyone to describe or locate a book, simply by knowing its ISBN.

Assignment

Your program will gather, record, and return a collection of book titles, each identified by the book's ISBN. Your program will feature a very simple menu that allows the user to perform any one of four operations, one at a time, repeatedly until the user chooses to quit the program. When prompted, the user may select one of the four operations by inputting a unique letter corresponding to one of the operations. The four operations (and their unique letters) are:

1. **[r] Record a Book:** This operation asks the user for two pieces of information, an ISBN and a book title. Your program will record the book's title and associate it to the given ISBN. Use a dictionary for this. You should assume that the ISBN is unique, and if a book was previously recorded using this ISBN, simply disregard the old record and replace it with the new one.

2. **[f] Find a Book:** This operation asks the user for one piece of information: an ISBN. Your program will attempt to find an existing record with this ISBN. If found, print the book title associated to this ISBN. If not found, print a message indicating that no such book could be found.

3. **[l] List all Books:** This operation does not ask the user for any additional information. Your program will immediately print a list of all known ISBN records, one per line. Each line should contain the ISBN followed by the book title that is associated to the ISBN.

4. **[q] Quit:** When this operation is selected, your program will print a friendly goodbye message to the user and then terminate.

5. **Something else?** If the user enters a different letter or phrase, print a message informing the user that this option is invalid, and allow them to try again.

Extra Challenges

- Add a bit of friendliness to your program! When your program first starts, before the menu options are first printed, display a brief message explaining your program and how to use it.

- Because this assignment does not require you to save the ISBN records to a file, you may assume that all records will be lost when the program quits. However, wouldn't it be useful if the records were *not* lost? Consider upgrading your program to write all records to a file just before the program quits, and then also load any records from the same file when the program is started again.

Hints

- Before starting, practice using dictionaries. You'll need to know how to create a dictionary, set a value by key, retrieve a value by key, check if a key exists and iterate through all keys and their values.

Sample

Program execution:

```
What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: r
Enter an ISBN: 978-0439708180
Enter a book title: Harry Potter and the Sorcerer's Stone
Book saved!

What would you like to do?
[r] Record a Book
```

```
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: r
Enter an ISBN: 978-0439023528
Enter a book title: The Hunger Games
Book saved!

What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: f
Enter an ISBN: 978-0439023528
Book found: The Hunger Games

What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: f
Enter an ISBN: 978-0439708180
Book found: Harry Potter and the Sorcerer's Stone

What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: l
1) 978-0439023528: The Hunger Games
2) 978-0439708180: Harry Potter and the Sorcerer's Stone

What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: x
Sorry, that option is invalid.

What would you like to do?
[r] Record a Book
[f] Find a Book
[l] List all Books
[q] Quit
Enter an option: q
Bye! See you next time!
```

Instructions

Create your program in the file `isbn_index.py`. Unit tests are available for download.

Your assignment is to create the following functions. The functionality for each function is described below. You must follow the specifications exactly, but may choose your own method for solving the problem described for each. Once you have completed a function you should run the unittest for that function and have it pass all tests. Fix any errors, warnings, and/or failures.

Because of the user input and output, not all functions are easily tested with unit tests. Be sure to test these functions by running your program and observing the correct behavior.

- `createIndex`
- `recordBook`
- `findBook`
- `listBooks`
- `formatMenu`
- `formatMenuPrompt`
- `getUserChoice`
- `getISBN`

- `getTitle`
- `recordBookAction`
- `findBookAction`
- `listBooksAction`
- `quitAction`
- `applyAction`
- `main`

`createIndex`

The function `createIndex` does not receive any parameters. It must return an empty dictionary for the index.

```
createIndex() -> { }
```

`recordBook`

The function `recordBook` receives three parameters, the index dictionary, the ISBN of a book, and the title of a book. Both the ISBN and title are strings. It must assign the title to the ISBN in the index. It does not return anything. However, the dictionary will be modified as a result of this function's work.

```
recordBook( index, "978-0439023528", "The Hunger Games" ) -> index should now contain the book
```

`findBook`

The function `findBook` receives two parameters, the index dictionary and the ISBN of a book. The ISBN is a string. The function returns the title of the book with the matching ISBN, if it exists. If the ISBN is not in the dictionary, then it returns the empty string.

```
findBook( index, "978-0439023528" ) -> "The Hunger Games"
findBook( index, "888-8888888888" ) -> ""
```

`listBooks`

The function `listBooks` receives one parameter, the index dictionary. The function returns a list of strings. Each string in the list is a line that shows a sequence number, the ISBN and the title of a book. See the examples for the format of the lines. If there are no books in the index, this function returns an empty list.

```
listBooks( empty_index ) -> [ ]
listBooks( index ) -> [ "1) 978-0439023528: The Hunger Games", "2) 978-0439708180: Harry Potter and the
Sorcerer's Stone" ]
```

`formatMenu`

The function `formatMenu` does not receive any parameters. It must return a list of strings that contains the lines of the menu.

```
formatMenu() -> [ 'What would you like to do?', '[r] Record a Book', '[f] Find a Book', '[l] List all
Books', '[q] Quit' ]
```

`formatMenuPrompt`

The function `formatMenuPrompt` does not receive any parameters. It must return a string that contains the prompt to ask the user which menu option they would like to select.

```
formatMenuPrompt() -> 'Enter an option: '
```

`getUserChoice`

The function `getUserChoice` receives one parameter, a string that contains a prompt for input. It must return a string that contains the text input by the user, with any leanding and trailing whitespace removed. If the

user gives an empty string, prompt them again, until they give a non-empty string. Note that this function interacts with the user, so there will be output to the screen and input from the keyboard when it is called.

```
getUserChoice( "Hello? " ) -> 'some text'
getUserChoice( "Choose" ) -> 'othertext'
```

`getISBN`

The function `getISBN` does not receive any parameters. It must prompt the user for an ISBN, and return the ISBN input by the user. The user's response must not have any leading or trailing whitespace. It must repeatedly ask the user for an ISBN, until the user gives a non-empty response. Note, you should probably call `getUserChoice` as part of this function.

```
getISBN( ) -> '978-0439708180'
getISBN( ) -> 'user-typed-this'
```

`getTitle`

The function `getTitle` does not receive any parameters. It must prompt the user for a book title, and return the title input by the user. The user's response must not have any leading or trailing whitespace. It must repeatedly ask the user for a title, until the user gives a non-empty response. Note, you should probably call `getUserChoice` as part of this function.

```
getTitle( ) -> "Harry Potter and the Sorcerer's Stone"
getTitle( ) -> 'user-typed-this too.  Weird!'
```

`recordBookAction`

The function `recordBookAction` receives the index dictionary as a parameter. It must ask the user for the ISBN and title of a book, and add it to the dictionary. This function does not return anything. However, it has the side effect of adding an entry to the dictionary. It also interacts with the user through input and output. Note you should be using some of the above functions to complete this function.

```
recordBookAction( index ) -> the index dictionary should be modified
```

`findBookAction`

The function `findBookAction` receives the index dictionary as a parameter. It must ask the user for the ISBN a book. If the book exists in the dictionary, it will display the book. If the book does not exist in the dictionary, it will give the user a message to let them know. The function does not return anything, and should not change the index.

```
findBookAction( index ) -> the index dictionary should not be modified
```

`listBooksAction`

The function `listBooksAction` receives the index dictionary as a parameter. It will display all of the books in the dictionary in the format shown in the examples. If there are no books in the dictionary, it must display a message to inform the user. The function does not return anything. The function must not change the dictionary.

```
listBooksAction( index ) -> the index dictionary should not be modified
```

`quitAction`

The function `quitAction` receives the index dictionary as a parameter. This function will display a message to the user indicating the end of the program. It will then terminate the program using `sys.exit( 0 )`. Be sure to do the correct `import` statement. This function does not return anything.

```
quitAction( index ) -> the program will end
```

`applyAction`

The function `applyAction` receives the index dictionary and a choice string as parameters. This function will call the appropriate action function based on the choice string. If the choice string does not match any accepted choices, it will display a message to the user. This function does not return anything. The dictionary may be changed as a result of the chosen action.

```
applyAction( index, "r" ) -> the dictionary will have a new book added.
applyAction( index, "f" ) -> the user will select a book to display.
applyAction( index, "l" ) -> the contents of the dictionary will be displayed.
applyAction( index, "q" ) -> the program will terminate.
applyAction( index, "bad-choice" ) -> the user will receive a message.
```

`main`

The function `main` receives no parameters, and returns nothing. This function ties everything together. Creating an index, repeatedly asking the user their choice and taking action.

```
main( ) -> the program runs.
```

Finishing Up

Lastly add this snippet at the bottom of your file which will execute your `main()` function when you run `isbn_index.py` but will allow it to be imported into the unittest files without executing the main function.

```
if __name__ == '__main__':
    main()
```

Pass-off instructions

1. To pass off this assignment you need to show your completed program to the lab assistants.
    - Show them your `isbn_index.py` code
    - Run `test_all.py` - **All tests MUST pass!**
    - Run `isbn_index.py`
    - *The lab assistant may additional tests they want you to run*
2. Upload your `isbn_index.py` file to canvas, please add a comment to the top of the file with your name and time your class meets.