ICS 31, Summer Session 10-wk, 2017

Lab 3

You will write a simple book management system. Your program will let the user enter information about a set of books and then issue commands to view information about the books.

You will implement a simple shell program which will allow the user to enter commands. Your program should print out a prompt to indicate that the user can enter a command. Your prompt should be a '$' character. At the prompt the user will type a command followed by a set of arguments for the command, separated by commas. Your program will perform whatever action is indicated by the command, and then your program will print a '$' character on a new line in order to let the user know that he/she can type in a new command.

Your program should accept the following commands.

1. **addbook**: This command indicates that the user wants to enter the information about a new book into the system. This command should be followed by three arguments on the line: a book name, the year of publication, and the author's name. This command adds the information about the new book into the system.

2. **printbooks**: This command prints the information about all books which have been entered into the system. The information about each book is printed on a separate line. Each line should have three pieces of information on it: a book name, the year of publication, and the author's name.

3. **deletebook**: This command indicates that the user wants to delete the information about a book from the system. This command should be followed by one argument, the name of the book.

4. **findbook**: This command searches the system for a book with a certain name. The command should be followed by one argument, the name of the book. If a book with the given name is in the system, the information about the book should be printed on a single line: the book name, the year of publication, and the author's name.

5. **findauthor**: This command searches for all books with a certain author. The command should be followed by one argument, the name of the author. For each book in the system with the given author, the book's information should be printed on a single line: the book name, the year of publication, and the author's name.

6. **findyear**: This command searches for all books with a certain publication year. The command should be followed by one argument, the year of publication. For each book in the system with the given publication year, the book's information should be printed on a single line: the book name, the year of publication, and the author's name.

7. **quit**: This command should end the program

Example Output

The following is an example of how your program should respond to commands. The text typed by the user is in bold.

```
$ addbook, book 1, 1995,  joe smith
$ addbook, book 2, 2000, joe smith
$ addbook, book 3, 1995, jane doe
$ addbook, book 4, 1996, jimmy dean
$ printbooks
book 1, 1995,  joe smith
book 2, 2000, joe smith
book 3, 1995, jane doe
book 4, 1996, jimmy dean
$ deletebook, book 4
$ printbooks
book 1, 1995,  joe smith
book 2, 2000, joe smith
book 3, 1995, jane doe
$  findbook, book 2
book 2, 2000, joe smith
$  findauthor, joe smith
book 1, 1995,  joe smith
book 2, 2000, joe smith
$ findyear, 1995
book 1, 1995,  joe smith
book 3, 1995, jane doe
$ quit
```

Additional Details

1. Most of the commands accept a number of arguments such as book title, year of publication, etc. Each argument may contain spaces inside them and your program must accept these spaces. For example, the command **`addbook, title 1, 1995, joe`** includes a space in the book title, "**`title 1`**". Notice that the title "**`title 1`**" is different from the title "**`title1`**" which contains no space.

2. The user must separate arguments by commas on the command line. The user might add any number of blank spaces to the left and right of the argument and your program must be able to accept that and operate correctly. For example, the command **`addbook, title 1, 1995, joe`** should be accepted and perform the same operation as the command **`addbook,  title 1,  1995, joe`** which has many additional spaces between the arguments.

3. The commands and arguments should be case-sensitive. For example, **`deletebook title1`** is different from **`deletebook TITLE1`**, and the command **`DELETEBOOK title1`** is unknown.

4. If the user enters a command which is not valid, your program should not crash. Instead, your program should ignore the command without changing the state of the book database, and print a new prompt for the user. Here are some examples:

    a. If the user types **`dbook title1`** then the command should simply be ignored and a new prompt should be printed because **`dbook`** is not a valid command.
    b. If the user types **`deletebook title1, title2`** the entire command should be ignored and no books should be deleted because the **`deletebook`** command does not accept two arguments.