

During the semester you will work in two applications to apply the data structures that we will cover in class. It is required to use Python 3.8 or higher. The applications are:

1. Calculator: It accepts a mathematical expression and the value of the variables and returns the result of the evaluations.
2. Basic book store system: It uses a book catalog and a shopping cart. The system will allow to search books and add them to the shopping cart and later remove from the shopping cart. The main database is the file books.txt that contains thousand of books and dvd titles. Each title corresponds to a single row with five register separated by ^ as follows:

key^title^category^rank^similar

where *rank* is the number of sold copies and *similar* is the list of books that are similar (starting with the size of the list followed by the key of the similar books.) For example:

```
0827229534^Patterns of Preaching: A Sermon Sampler^Book^396585^5 0804215715 156101074X 0687023955 0687074231 082721619X
```

The entry with key 0827229534 has the title "Patterns of Preaching: A Sermon Sampler" is a book that has been sold 396585 copies and has 5 books that are similar to it 0804215715 156101074X 0687023955 0687074231 082721619X.

A template with a preliminary implementation is provided in BeachBoard (template.zip) under *Template* of the *Content Tab*. The template provides the classes and a simple menu to interact. Modify the menu to support the required functionalities. Read the readme.txt file for the documentation.

Note Only assignments that use the template will be graded.

LAB 1: ARRAY-BASED LIST

Learning objectives: CLO 1, CLO 3, CLO 4

Use Python 3.8 or higher for the assignment:

1. Implement ArrayStack, ArrayQueue and ArrayList covered in the lecture 2 (ArrayBased Lists). All the data structures should be fully functional and must follow the logic presented in the lecture.

Learning objectives: CLO 1, CLO 3

Test your data structures using the following tests.

- Remove one element from an empty Stack, Queue, List
- Stack: Add 5 elements and remove them checking that they are in opposite order of insertion, e.g., Inserting the sequence 5,4,3,2,1 result in the sequence 1,2,3,4,5 when removing
- Queue: Add 5 elements and remove them checking that they are in the same order of insertion, e.g., Inserting the sequence 1,2,3,4,5 result in the sequence 1,2,3,4,5 when removing

- List: Add 5 elements in different positions (including the first and last) and check that they are in order, e.g., *add*(0, 4), *add*(0, 1), *add*(1, 3), *add*(1, 2), and *add*(4, 5). Then *get*(*i*) should return *i* + 1. Remove 2 elements, e.g., index 2 and 3 and the final array should be "1,2,4".
2. Implement a *RandomQueue* to randomly remove the books. This is an implementation of the Queue interface in which the *remove*() operation removes an element that is chosen uniformly at random among all the elements currently in the queue. The *add*(*x*) and *remove*() operations in a *RandomQueue* should run in amortized constant time per operation.

Learning objectives: CLO 3

Hint: Use the random methods in your language *randint* from the module random.

Test your *RandomQueue* using the following tests.

- Remove one element from an empty *RandomQueue*
- Add 5 elements and remove all. Check that the remove operation return random values

3. Calculator:

Learning objectives: CLO 1

- (a) A *mathematical expression* is a sequence of numbers, letters and "(", ")" characters that are properly matched. For example, "a + (b*c + d) / (a-c)" is a matched expression, but "a + (b*c + d / (a-c)" is not. Use your Stack implementation (*ArrayStack*) to decide whether a user expression is valid in $O(n)$ time.

Hint: Consider the Invariant that at every closing parenthesis there must be one open parenthesis.

The menu should give an option in the menu to introduce expressions dynamical.

Test your program:

- An empty expression.
- An expression with more ")" than n "(" , e.g., "(a+b*)c -d)".
- An expression with more "(" than n ")". e.g., "(a+b*(c -d)".
- An expression with same number of "(" and ")" but not matched, e.g., ")a+b*(c -d)".
- An expression without parenthesis, e.g., "a+b*c-d"
- An expression with matched parenthesis, e.g., "(a+b)*(c -d)"

4. Book Store System:

Learning objectives: CLO 1, CLO 3, CLO4

- (a) Load the catalog "books.txt" in an instance *bookCatalog* of your *ArrayList*. Each row in books.txt is a book that we store in a node (class). Thus, *bookCatalog* stores a list of the class *Book*. The template implements this option that will be fully functional when the *ArrayList* is implemented. In development time, use the file "booktest.txt" with few books. Once you think it is ready, use the main file "books.txt".
- (b) Create an instance *shoppingCart* of *ArrayQueue*. Initially, it is empty.
- Adding a book by index: The user selects the index *i* to add of the book in *bookCatalog*. That is, add the *Book* to *shoppingCart* that *bookCatalog.get(i)* returns. The template implements this option that will be fully functional when your *ArrayQueue* is completed. Observe that this operation emulates adding a book to a shopping cart in any online store.

For loop here

- ii. Remove a book from *shoppingCart*: Use the remove method of the queue. The template implements this option that will be fully functional when your ArrayQueue is completed.

- iii. Search a book by title: Given an infix (phrase) by the user, display the title and index of all the books in *bookCatalog* that contains the infix. The search should be case sensitive, i.e., capital letter and lower case letters are not the same. Add an option in the menu and let the user introduce the infix.

Hint: Use a for loop and the *in* operator to test whether the prefix matches

- iv. Create a RadomQueue instance *randomShoppingCart* to randomly remove books from the shopping cart. Add the options in the menu to add books by index and remove books from *randomShoppingCart*

Test your program:

- Remove a book from an empty shoppingCart.
- Add books with index 0, 542683, 271341, 135670, 407012 to *shoppingCart*:
- Remove all books in *shoppingCart* using the method *removeBooksFromCart*. They should return:

Book in shopping cart

Book: 0827229534

Title: Patterns of Preaching: A Sermon Sampler

Group: Book

Rank: 396585

Similar: 5 0804215715 156101074X 0687023955 0687074231 082721619X

Book in shopping cart

Book: B00005MHUG

Title: That Travelin' Two-Beat/Sings the Great Country Hits

Group: Music

Rank: 0

Similar: 5 B000080ETQ B0000506KL B00006RY87 B00020TI98 B0000634HG

Book in shopping cart

Book: 055329315X

Title: Reckless

Group: Book

Rank: 92964

Similar: 5 0553561537 0553293168 0553289322 0553283545 0553293176

Book in shopping cart

Book: 1841121495

Title: Big Shots: Business the Richard Branson Way

Group: Book

Rank: 464292

Similar: 5 0812932293 0582512247 0684865165 0761503439 0471196533

Book in shopping cart

Book: 1571686223

Title: The Letters of John Wesley Hardin

Group: Book
 Rank: 1041036
 Similar: 0

World of P

- Searching for books by at least the following prefix:
 - (a) Empty prefix.
 - (b) "Word of P" should display 10 books
 - (c) "Tears of the" should display 8 books.
- Add books with index 0, 542683, 271341, 135670, 407012 to *randomShoppingCart*.
- Remove all books in *randomShoppingCart*. It should return the list without any order.

Submit all the source code (Python files (.py) in a zip file. The name of the zip file with the source code must be your first name, second name, and the lecture title separated by a hyphen. For example, oscar-ponce-arraybase.zip

Submissions that do not follow the previous specification will be rejected and you will have 0 in the lab.

RUBRICS

	Level 4 2 Pt	Level 3 1.5 Pt	Level 2 1 Pt	Level 1 0.5 Pt
ArrayStack implementation	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
ArrayQueue implementation	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
ArrayList implementation	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
RandomQueue implementation	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
Validating mathematical expression	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
Searching books by infix	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete
Random Shopping Cart (menu options and functionality in BookStore)	It is always correct without crashes	It is always correct and eventually it crashes	It is not always correct and eventually it crashes	It is not correct or incomplete