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 2: LINKED-LIST

Learning objectives: CLO 1, CLO 3, CLO 4
Use Python 3.8 or higher for the assignment:

1. Implement SLLStack, SLLQueue and DLList covered in the lecture 3 (Linked-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 and check 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 and check 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 list should be "1,2,5".

2. Write a DLList method $is\_palindrome()$ that returns $true$ if the list is a *palindrome*, i.e., the element at position i is equal to the element at position $n-i-1$ for all $i \in \{0, \ldots n-1\}$. Your code should run in $O(n)$ time.

   Hint: Traverse the list forward and backward simultaneously.

   The menu should give an option to test palindromes.

   Learning objectives: CLO 3

   Test your program:

   - An empty word. It should return true
   - A palindrome word with one letter, e.g., "a"
   - A palindrome word of even length, e.g., "hannah"
   - A palindrome word of even length, e.g., "eve"
   - A word that is not palindrome.

3. Write a DLList method $reverse()$ that reverse the whole list. Your code should run in $O(n)$ time.

   Hint: Consider modifying the references (next and pred)

   The menu should give an option to reverse the list.

   Learning objectives: CLO 3

   Test your program:

   - An empty list.
   - A list with one element.
   - A list with n elements. For example, "5,4,3,2,1" should return "1,2,3,4,5"

4. Design and implement a MaxQueue data structure that can store comparable elements and supports the Queue operations $add(x)$, $remove()$, and $size()$, a as well as the $max()$ operation, which returns the maximum value currently stored in the data structure. **All operations should run in constant time**. MaxQueue is a specialization of the SLLQueue. Therefore, it inheritances all the methods from SLLQueue and overwrites add and remove operations. The template provides the inheritance.

   Learning objectives: CLO 3

   Hint: Consider using a second queue that at adding it adds the maximum between the last value and the new value.

   Test your MaxQueue implementation using:

   - Remove from an empty MaxQueue
   - Add 5 elements to MaxQueue: $push(3), push(1), push(4)$ and $push(2)$.
   - Check that $max()$ returns 4.
   - Remove 2 elements: $remove()$ that should return 3 and 1 in that order.
   - Check that $max()$ returns 4.
   - Remove one element: $remove()$ that returns 4.
   - Check that $max()$ returns 2.

5. Book Store System: This part of the assignment is identical to Lab 1 except that $bookCatalog$ is an instance of your DLList and $shoppingCart$ is an instance of your SLLQueue. In development time, use the file "booktest.txt" with few books. Once you think it is ready, use the main file "books.txt".

   Learning objectives: CLO 1, CLO 3, CLO 4

   (a) Load the catalog "books.txt" in an instance $bookCatalog$ of your **DLList**. Each row in books.txt is a book that we store in a node. Thus, $bookCatalog$ stores a list of the class Books. This part of the assignment is identical of the Lab 1 except that $bookCatalog$ is an instance of your DLList.

   (b) Create an instance $shoppingCart$ of SLLQueue. This part of the assignment is identical of the Lab 1 except that $shoppingCart$ is an instance of your SLLQueue.

      i. 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 SLLQueue is completed. This part of the assignment is identical to Lab 1.

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 SLLQueue is completed. This part of the assignment is identical to Lab 1.

iii. Search books 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. Reverse the order of the shopping cart using your reverse implementation. Add an option in the menu.

v. Create an instance $bestSelling$ of $MaxQueue$ and add all the book in the shopping cart. We overwrite the operator ($>$ greater than) in the class $Book$ to compare Books. The template already implements it. Add an option in the menu to print the best selling book.

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  B00008OETQ  B00005O6KL  B00006RY87  B0002OTI98  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
```

- Add books with index $0, 542683, 271341, 135670, 407012$ to $randomShoppingCart$:
- Remove all books in $randomShoppingCart$. It should return the list without any order.
- 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.

6. What is the advantage and disadvantage of Linked List compared with Array-Based data structures.

   Learning objectives: CLO 4

   Hint: Compare the time to access an element i in the list. Recall that every millisecond count.

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-linkedlist.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
SLLStack implementation	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
SLLQueue implementation	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
DLList implementation	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
Palindrome test	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
Reverse DLList	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
MaxQueue implementation	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
Searching books by infix	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
Reverse and MaxQueueuse (menu options and functionality in BookStore)	It is always correct without crashes	Eventually it crashes or return incorrect results	It frequently crashes and/or return incorrect results	It is not correct or incomplete
Answer to Question 6	N/A	N/A	Correct	Incorrect