|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***MODULE NUMBER*** | | 05 | | | |
| ***TOPIC(S)*** | | Linked Lists (Single and Double) | | | |
|  | | | | | |
| **--- PRECLASS ---** | | | | | |
| **SHORT VIDEO** | <https://goo.gl/CUz8r4>  (Singly linked lists: focus on ADT definition, not on the C code) | | | |  |
| <https://goo.gl/5gC0e4>  (Doubly linked lists: again, do not focus on the C code) | | | |  |
| **BOOK** | GTG Chapter 7 - Sections 7.1 and 7.3 (do not read the sub-sections 7.1.1 etc. and 7.3.1 etc.) | | | |  |
| **QUIZ** | <https://goo.gl/forms/rQxsgNTz216SB7043> | | | |  |
| **TO-DO** | PyCharm: check the implementation of the classes LinkedList and DoublyLinkedList in module05 >>> preclass  (Note that the implementation differs slightly from the one in the book chapter 7) | | | |  |
| **--- PROBLEM SET 1 (Thursday Sept 29th) ---**  - Complete the functions in module05 >> part01  **find(L,e)** that looks for an element e in a singly linked list L. The method prints the position of e if e belongs to the list and “Element not found” otherwise.  (Assume no duplicates in the list, that is, all elements in the list are unique).  **cat(L1, L2)** that concatenates the list L2 at the end of L1.  **copy\_and\_cat(L1,L2)** returning a list, which is the concatenation of L1 and L2.  (Suggestion 1: you should make copies of the lists L1 and L2 and then “cat” the copies using the method created at the previous step)  Suggestion 2: to make a copy of a list, note that while you can read the element of a list sequentially, you can only insert nodes at the start of the copy, so…)  Write a recursive algorithm **len\_recursive(L)** that returns the number of nodes in a singly linked list L | | | | | |
| **--- POST-CLASS 1 ---** | | | | | |
| **TO-DO** | Complete problem set 1 | | | |  |
| **--- PROBLEM SET 2 (Tuesday October 4th) ---**  Complete the fnctions in module05 >>> part02  **swap\_first\_last(DL)** that swaps the first and last node in a doubly linked list DL  **insert\_at\_pos(DL, e,n)** that inserts an element e at n-th position in a doubly linked list DL (try to think of the fastest possible algorithm for doing this…)  **special\_copy(L1,L2)** returning a new doubly linked list which contains the same element of DL1 concatenated with the elements of DL2 in reverse order | | | | | |
| **POST-CLASS 2** | | | | | |
| Complete problem set 2 | | |  |  |  |