CMPUT 175

Lab 7: Doubly linked list

Lab 7 Exercises

- Keep in mind that all class variables must be private.
- Test your code by using the corresponding test function in the files given to you
- For the doubly linked list you can see the slides on eclass or search wikipedia for doubly linked list if you need an explanation on how it works

Exercise 1: Double Linked List

- The goal of this exercise is to create a doubly linked list. You have to use the file d_linked_node.py and the d_linked_list.py
- The d_linked_node is ready and the d_linked_list you will have to complete in the same way you did in other labs.
- The methods that you saw in class about d_linked_list are already implemented
- Consider that element and item is the value on the d_linked_node and not the next and previous

Exercise 1: Methods

- add(self, item)
  - Adds a new item to the front of the list
- remove(self,item)
  - Removes the first element that is equal to item from the list. If that doesn’t exist it changes nothing
- append(self, item)
  - Adds item to the end of the list
- insert(self, pos, item)
  - Add item to the given position in the list (i.e if pos is 0 then item goes to the beginning of the list)

Exercise 1: Methods

- pop(self, pos=None)
  - This function gets an optional argument for position
  - If the position has been given, removes and returns the item in the given position
  - If the position has the default value(None), removes and returns the last item in the list
- search_larger(self, item)
  - Returns the position of the first item that is larger than item, -1 if there is no item larger
- get_size(self)
  - Returns the size of the list

Exercise 1: Methods

- get_item(self, pos)
  - Returns the item that exists at pos, Raises exception if pos doesn't exist
  - pos can be negative which means the position from the end of the list. (-1 is the last item, -2 the second from last etc.)
- __str__(self)
  - Returns a string that is the elements of the DLinkedList with spaces in between

Exercise 1: Methods already implemented

- search(self, item)
  - Returns true if the item is inside the list false otherwise
- index(self,item)
  - Returns the index of the item or -1 if the item is not in the list

Exercise 2: Maybe Sorted List

- For this exercise you have to use the d_linked_list that you created in exercise 1.
- You have to implement the methods for the class m_sorted_list, which when initialized takes a parameter sorted which is true or false. When the parameter is true the list is sorted in ascending order. When the parameter is false it acts as a normal unbounded queue

Maybe Sorted List: Methods

- __init__(self, m_sorted)
  - Initialization of the class with parameter mSorted which is true or false
- add(self, item)
  - Adds the item to the correct position if the list is sorted, or adds the item to the end of the list if the list is not sorted

Maybe Sorted List: Methods

- pop(self)
  - Removes and returns the first item in the list if the list is not sorted, or returns and removes the largest number in the list if it is sorted
- search(self, item)
  - Returns a tuple, with the first element being true or false if the item was found in the list or not, and the second element is the index of the of the item if it was found, or if it is not found, if the list is sorted it is the index of the first item that is larger than item, and -1 if the list is unsorted or there is no larger item in the list

Maybe Sorted List: Methods

- change_sorted(self)
  - Change the list from sorted to unsorted, or raises an exception with the message “I don’t know how to sort a doubly linked list yet”
- get_size(self)
  - Returns the size of the list
- get_item(self,pos)
  - Returns item at pos
- __str__(self)
  - Returns a string that is the elements of the list