University of Waterloo
CS 234 - Data Types and Structures
Spring 2014
Assignment 4
Due Wednesday July 9th at 4:00 pm

Assignment Guidelines

- Clarifications about the assignment will be posted on Piazza in the thread "Assignment 4 [Official]."
- Instructions on how to submit your assignments to MarkUs will also be posted in Piazza. Your programs must work in the linux.student.cs.uwaterloo.ca environment using Python 2.7.3.
- Use A4-coversheet.pdf for the first page of your assignment or format the top of the first page of your assignment in *exactly* the same manner.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.

Question 1: Linked List [30 marks]

For this question you will implement a singly linked list data type with a reference to both the head and the tail of the linked list. Rather than implement a data structure all at once, for this question, you will implement some functions of the linked list, test those function then implement and test some more. Using this approach you would typically want to implement functions like *str*() and *len*() early on so that you inspect the list and consequently catch bugs early. Also note that although this class is called a LinkedList it is more specifically a singly linked list.

a) Implement *append*(), *str*() and *len*() function for the (singly) LinkedList class by modifying the file A4Q1.py provided on Learn. Here I have provided the constructors for the ListNode and LinkedList classes, however the *append*() method does nothing and *str*() and *len*() command always return a blank string and zero respectively. Modify this file so that these last three functions are implemented and call the modified file A4Q1a.py

   The *str*() function for the LinkList class should create exactly the same string format as a Python list. For example, if the LinkedList is empty, the string should be "[]" and if the LinkedList contains the items 1 and 2 the resulting string representation would be "[1, 2]". The head is on the left and the tail is on the right.

   Question 1a Deliverables: On MarkUs, submit a file called A4Q1a.py which contains the original (i.e. unmodified) constructors for the ListNode and LinkedList class and an implementation of the *append*(), *str*() and *len*() functions.

b) Create a file called A4Q1Test.py that contains a function called *main*() which takes an integer parameter, *testLevel*, such that a call to the function main(1) would thoroughly test the constructor as well as the *append*(), *str*() and *len*() functions of the LinkedList class of A4Q1a.py.

   Question 1b Deliverables: See part f).

c) Copy the file A4Q1a.py to a new file A4Q1c.py and in this new file also add the function *remove*(*anItem*) to the class LinkedList. This function searches through the Linked List (starting at the *head*) and removes the first occurrence of *anItem* or raises a ValueError if *anItem* is not found.

   Question 1c Deliverables: On MarkUs, submit a file called A4Q1c.py which contains all of A4Q1a.py and the function *remove*(*anItem*).

d) Modify the file called A4Q1Test.py so that when function *main*() with the parameter 2, i.e. *main*(2), it will do all the tests that a call to *main*(1) will do plus it will thoroughly test the function *remove*(*anItem*).

   Question 1d Deliverables: See part f).

e) Copy the file A4Q1c.py to a new file A4Q1e.py and in this file also add the following functions.

pop(i) removes and returns the i^{th} item from the Linked List. Here *i* must be non-negative and assume the elements are enumerated starting at zero (just like strings and Python lists). If *i* is too large raise an IndexError.

insert(i, anItem) inserts *anItem* into the Linked List at position *i*. Again like pop, *i* must be non-negative and the elements are enumerated starting at zero. If *i* is too large raise an IndexError.

count(anItem) returns the number of times that *anItem* occurs in the Linked List. If anItem does not occur return zero.

Question 1e Deliverables: On MarkUs, submit a file called A4Q1e.py which contains all of A4Q1c.py and the Link List class functions *pop*(), *insert*() and *count*().

f) Modify the file called A4Q1Test.py so that when function *main*() with the parameter 3, i.e. *main*(3), it will do all the tests that a call to *main*(2) will do plus it will thoroughly test the function *pop*(), *insert*() and *remove*(). Your code for parts b), d) and f) will be visually assessed for thoroughness, organization, clarity, and meaningful error messages [**10 marks**].

Question 1b,d, f Deliverables: On MarkUs, submit a single file called A4Q1Test.py which contains a function *main*(*testLevel*).

g) Your code for parts a), c) and e) will be visually assessed for simplicity, efficiency and clarity (i.e. organization, well documented class and function headers, identified preconditions and postconditions, meaningful error message and variable names) [**10 marks**].

h) You code will also be assess via automated testing using test cases that we have created [**10 marks**].

Note that for this question you made three separate files A4Q1a.py, A4Q1c.py, and A4Q1e.py that progressively implemented the LinkedList class. In "real life" you would just create one file, but you would add on to it in a few steps making sure your initial functions are working (through testing) before added more functions. Typically you would use this approach when dealing with an application area or a set of python library functions that you are unfamiliar with.

Question 2: Course Project [20 marks]

This part of the assignment is not finalize yet, but will be finished soon.