Practical Questions

CSCA48 – Week 4

Question #1

a) Implement a LinkedList class, with all the methods described in class. We know you have the code already, but make sure you can do it on your own without looking at Nick's code

b) Implement a stack using your linked list.

c) Implement a queue using your linked list.

d) Now change your LinkedList class implementation so that it doesn't use linked lists. Cheat and use python's built in lists. Don't change the ADT in any way, just the implementation.

e) Check that your stack/queue still work. If not, figure out where you broke your own abstraction.

f) Nick wants to implement his UnfairQueue using linked list. In order to keep the code in UnfairQueue nice and clean, a special linked list is needed, let's name it…. Umm. .. UnfairLinkedList just to make our life easier. It should have methods for bump_to_tail, bump_to_head and reverse.

g) Write a function that takes a linked list as input, and returns True if it contains a loop (i.e., if you would eventually access at least one node twice by following head.next.next.next…. **Challenge**: Do this twice, one way that is efficient in terms of the number of nodes that we have to check, and once that is efficient in terms of the amount of extra space required.

h) Make the singly linked list you implemented earlier into a doubly linked list (all nodes have a pointer to the previous node as well as the next). You will start this in tutorial if you need more clarification.

i) Write a function that takes a linked list as input and removes all duplicate values from that list (you can define 'duplicate', and decide which of the duplicates is removed in any way you wish)

j) **(Challenge)** Merging list

i. Write a function (not a method) that takes two sorted linked list and return a new linked list with the two lists merged.

ii. Write a method in a LinkedList class called merge. It takes one input which is another linked list and merge with itself. Notice that L1.merge(L2) modifies L1. You may assume both list are sorted

k) (Challenge)

Implement a class called Browser to simulate a web browser. It should contain at least the following 3 methods:

- display – print the current url
- back – move to the last url visited before visiting this page (if there's no previous url to visit, do nothing)
- forward – go forward along the list of backtracked urls
- visit – visit a new url (note that this should erase the history of our forward links)

l) **(Challenge)** Complete the UML diagram for the following

i. We need to represent books and authors. Authors should have a first and last name and a mailing address. Sometimes books change their titles and number of pages, and it we often need to add just one extra page, so it would be good to have a simple way of doing that. Given a book, I should be able to get all of its authors, and given an author I want to be able to get all the books they've written.

ii. Books can be paperback or hardcover, and hardcover books can either be regular or special edition. Paperbacks cost less, and a special edition's price depends on whether or not it has a certificate with it. The number of copies we print will also depend on the type of book, there's a formula for paperback based just on the number of pages, but for hardcover it depends on the author, and for special editions, we always print a fixed number.

iii. Authors are either contractors (who have an agent and are paid solely on the number of pages they've written), or salaried (who have an annual salary based on the number of years they've worked with the company). We need to be able to see for any given author how much they've been paid over the years

Discussion Questions

1. What do we mean when we say that python lists "don't really exist"? What is wrong with the way we have been imagining them up until now? What is unrealistic about the way we draw the memory model?
2. The Node class is often referred to as a private or hidden class? Some people will even write it as class _Node. What does this mean? How does this relate to private/hidden methods?
3. What is the benefit of drawing UML diagrams? Is it all just extra overhead, or does it really help? How does it fit with our policy that computer scientists should be "lazy"?

(Challenge) Logic Question

Nick and Brian are inventing a new board game. They play on a standard checkerboard. Each of them take turns putting a piece of their colour (Brian plays black, Nick plays red) down on one of the squares of the board. If any player puts down a piece such that it forms a square with 3 other pieces of the same colour, the game is over and that person loses.

After a while, they get bored of playing the normal way, and try to see how long they can keep the game going with neither of them losing. To add an extra twist, they decide to account for squares tilted at any angle.

Here's the question: What is the largest board such that Nick and Brian can fill the entire board without either of them losing? And what does that board look like? (Try it for both the "standard" game, where squares only count if they're vertically or horizontally aligned, as well as the "advanced" game where tilted squares count).