Doubly Linked Lists

CSCI 112, Lab 4

File names: Names of files, functions, and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code. Do not use code written by anyone else, in the class or from the internet.

Documentation: Each file should begin with a docstring that includes your name, the class number and name, the lab number, and a short description of the lab, as well as documentation pertinent to that particular file.

Doubly linked list: The linked list implementation given in the text is called a singly linked list because each node has a single reference to the next node in sequence. An alternative implementation is known as a doubly linked list. In this implementation, each node has a reference to the next node (commonly called next) as well as a reference to the preceding node (commonly called back). The head reference also contains two references, one to the first node in the linked list and one to the last.

Implement this data structure in Python. Follow the style of the singly-linked list in the text. Implement all the operations listed in §4.20. Call the file `doublelist.py`.

Write up an analysis of all of your operations and deduce their Big-O runtime, based on your code. Contrast each one with the runtime of the singly-linked list from the text. When, and if, they differ, say why one has a better runtime than the other.

At the end of your writeup, discuss why one or the other implementation would be better or worse, and in which situations. Title the writeup either `singledouble.txt` or `singledouble.tex`.

Zip all files in a folder called `csci312lab03yourname` and turn in on canvas