Problem Set 1

All parts are due on September 13, 2019 at 6PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `alg.mit.edu`.

Problem 1-1. [20 points] **Asymptotic behavior of functions**

For each of the following sets of five functions, order them so that if f_a appears before f_b in your sequence, then $f_a = O(f_b)$. If $f_a = O(f_b)$ and $f_b = O(f_a)$ (meaning f_a and f_b could appear in either order), indicate this by enclosing f_a and f_b in a set with curly braces. For example, if the functions are:

$$f_1 = n, \qquad f_2 = \sqrt{n}, \qquad f_3 = n + \sqrt{n},$$

the correct answers are $(f_2, \{f_1, f_3\})$ or $(f_2, \{f_3, f_1\})$.

Note: Recall that a^{b^c} means $a^{(b^c)}$, not $(a^b)^c$, and that $\log$ means $\log_2$ unless a different base is specified explicitly. Stirling's approximation may help for part **d)**.

a)	**b)**	**c)**	**d)**
$f_1 = (\log n)^{2019}$	$f_1 = \log((\log n)^{6006})$	$f_1 = 3^{4n}$	$f_1 = 2^n$
$f_2 = n^2 \log(n^{2019})$	$f_2 = \log\log n$	$f_2 = 3^{2^n}$	$f_2 = n^3$
$f_3 = n^3$	$f_3 = n^{6006} \log n$	$f_3 = 2^{2^{n+2}}$	$f_3 = \binom{n}{n/2}$
$f_4 = 2.019^n$	$f_4 = \log(6006^{n^{6006}})$	$f_4 = 4^{n^3}$	$f_4 = n!$
$f_5 = n \log n$	$f_5 = (\log n)^{\log(n^{6006})}$	$f_5 = 10^n$	$f_5 = \binom{n}{3}$

Problem 1-2. [25 points] **Better Sequences**

In Lecture 2, we used both linked lists and dynamic arrays to support the sequence interface. The linked list presented supports $O(1)$-time `insert_first(x)` and `delete_first()` operations, while the dynamic array supports amortized $O(1)$-time `insert_last(x)` and `delete_last()`. This seems like a compromise. In this problem, you will extend each of these data structures to support all four first/last sequence operations in $O(1)$ time. (For any algorithm you describe in this class, including data structure operations, you should **argue that it is correct**, and **argue its running time**.)

(a) [5 points] Describe how to modify a linked list to store and maintain $O(1)$ additional information to support the `insert_last(x)` operation in worst-case $O(1)$ time.

(b) [10 points] Describe how to modify a linked list to support each of the four first/last sequence operations in $O(1)$ time. Are your operation running times worst-case or amortized?
Hint: Store and maintain $O(1)$ additional information at each node.

(c) [10 points] Describe how to modify a dynamic array to support each of the four first/last sequence operations in $O(1)$ time. Are your operation running times worst-case or amortized?
Hint: A dynamic array has fast last operations because it maintains additional space at the end of the array. Can you generalize?

Problem 1-3. [10 points] Given a data structure that supports the four first/last sequence operations (`insert_first(x)`, `delete_first()`, `insert_last(x)`, `delete_last()`) each in $O(1)$ time, describe algorithms to implement the following higher-level operations in terms of the lower-level operations. Recall that `delete` operations return the deleted item. (Remember to argue **correctness** and **running time**!)

(a) [5 points] `shift_left(k)`: Move the first k items in order to the end of the sequence in $O(k)$ time. (At the end of the operation, the k^{th} item should be last and the $(k+1)^{\text{th}}$ item should be first.)

(b) [5 points] `swap_ends()`: Swap the first item in the list with the last item in the list in $O(1)$ time.

Problem 1-4. [45 points] **Jen & Berry's**

Jen drives her ice cream truck to Linkoln Elementary at recess. All the kids rush to line up in front of her truck. Jen is overwhelmed with the number of students (there are $2n$ of them), so she calls up her associate, Berry, to bring his ice cream truck to help her out. Berry soon arrives and parks at the other end of the line of students. He offers to sell to the last student in line, but the other students revolt in protest: "The last student was last! This is unfair!"

The students decide that the fairest way to remedy the situation would be to have the back half of the line (the n kids furthest from Jen) reverse their order and queue up at Berry's truck, so that the last kid in the original line becomes the last kid in Berry's line, with the $(n+1)^{\text{th}}$ kid in the original line becoming Berry's first customer.

(a) [20 points] Given a linked list containing the names of the $2n$ kids, in order of the original line formed in front of Jen's truck (where the first node contains the name of the first kid in line), describe a $O(n)$-time algorithm to modify the linked list to reverse the order of the last half of the list. Your algorithm should not make any new linked list nodes or instantiate any new non-constant-sized data structures during its operation. (Remember to argue **correctness** and **running time**!)

(b) [25 points] Write a Python function `reorder_students` that implements your algorithm. You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.

```
def reorder_students(L):
    '''
    Input:  L    | linked list with head L.head and size L.size
    Output: None |
    This function should modify list L to reverse its last half.
    Your solution should NOT instantiate:
        - any additional linked list nodes
        - any other non-constant-sized data structures
    '''
    ##################
    # YOUR CODE HERE #
    ##################
    return
```