# CS 1332 Recitation 2 Worksheet – Doubly Linked Lists, Stacks, Queues

CS 1332 TAs

January 20, 2019

This worksheet covers material from this week's recitation. It is meant to be additional exercise for your benefit and will not be graded. Feel free to collaborate with other students on this and ask TAs for help. Distribution of this worksheet is **not** permitted.

## 1 Questions

Note: Write the amortized complexity where appropriate and state that it is amortized. Assume all LL's have tail pointers.

1. `addToBack()` in a Doubly Linked List: _____
2. `removeFromBack()` in a Doubly Linked List: _____
3. `addToFront()` in a Doubly Linked List: _____
4. `push() and pop()` in an array-backed Stack, with elements added at the **back**: _____
5. `push()` in an SLL-backed Stack with elements added at the **back**: _____
6. `pop()` in an SLL-backed Stack with elements added at the **back**: _____

## 2 Scenario Matching

Match each scenario with the most appropriate data structure that could be used. Each choice may be used once, more than once, or not at all.

1. Apple Music's "Play Next" button lets a user create a list of songs to play. A user can click "Play Next" on Toto – Africa, and Africa will be played next. If a user were to click "Play Next" on Toto – Africa, Darude – Sandstorm, and Rebecca Black – Friday, in that order, then Friday will be played, followed by Sandstorm, followed by Africa. The most appropriate data structure for this list of songs is _____

2. Several scientists are sharing a massive server. They each have some jobs they would like to run on the server, but the server can only run one job at a time. If Dr. HB, Dr. Trilok, and Dr. Avimothy submitted jobs in that order, then the server would execute Dr. HB's job, Dr. Trilok's job, and Dr. Avimothy's job in that order. The most appropriate data structure for handling jobs is _____

3. Spotifly is a hip new music software competing with Apple Music. The engineers decided to have a "Play eventually" button. If a user clicked "Play eventually" on Nickelback – Photograph, Michael Dapaah – Man's Not Hot, Owl City – Fireflies, then Photograph, Man's Not Hot, and Fireflies will play in that order. The most appropriate data structure for this list of songs is _____

A. Queue

B. Stack

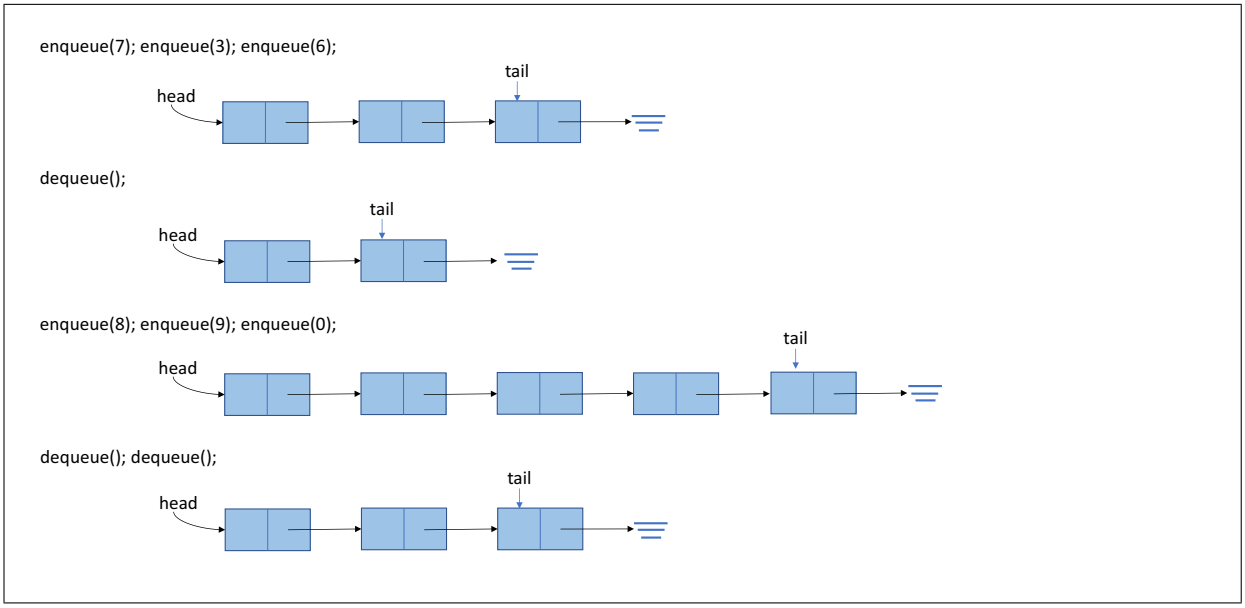
C. Array

4. An OS must run several processes one at a time. To do this, it iteratively gives each process 5ms. to run. Once process $N$ has run for 5ms., the OS cycles back to the beginning of the process list and gives 5ms. for process 1 to run. The most appropriate data structure for storing these processes is _____

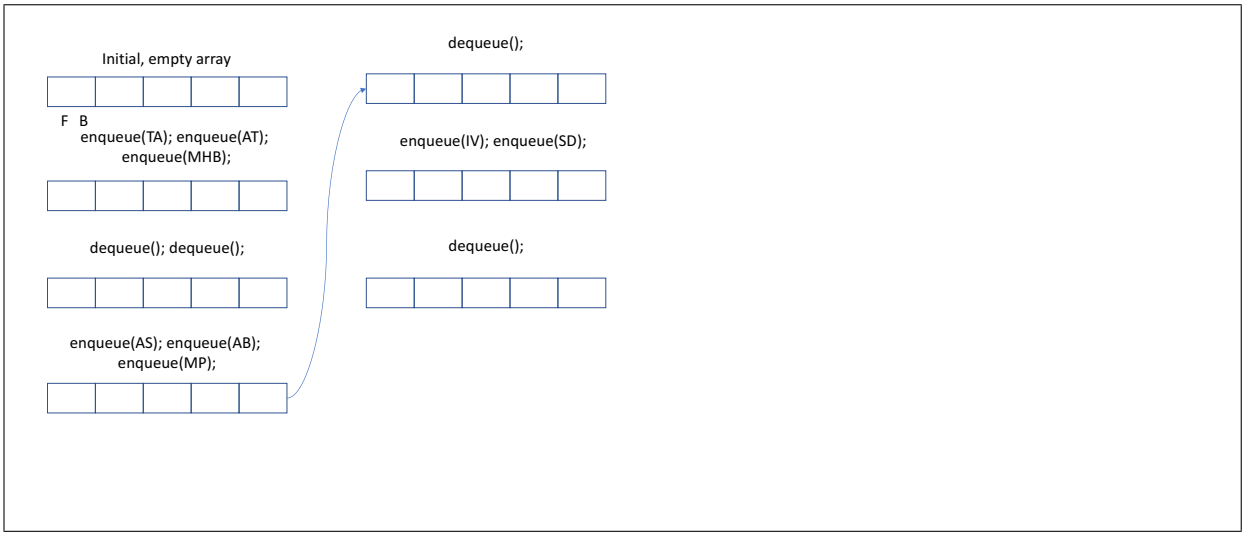
D. Circular Linked List

# 3 Diagramming

1. Fill in each node in this SLL-backed Queue with the correct data following the corresponding operation(s).




2. Fill in each entry in this Array-backed Queue with the correct data following the corresponding operation(s). You may leave null entries blank. Specify where *front* and *back* are pointing to by writing an 'F' or a 'B' next to the matching index.



# 4 Above and Beyond

This is a small set of challenge questions related to the material in the course, similar to those that you may find in a Software Engineering interview. Again, feel free to collaborate with other students or ask a TA for help.

1. Write a function that determines if a given *String*, which will only contain the characters '(', ')', '{', '}', '[', ']', is "valid". A valid string is defined to be one where 1) open brackets are closed by the same type of brackets, and 2) open brackets are closed in the correct order.

   Examples:
   Input: "()"
   Output: true

   Input: "(]"
   Output: false

   Input: "({}[])"
   Output: true

   Input: "([)]"
   Output: false

   Expected time: $O(n)$, Expected space: $O(n)$, where $n$ is the length of the input string.

2. Write a function that determines if a given a singly linked list is a palindrome.

   Examples:
   Input: $1 \rightarrow 2$
   Output: false

   Input: $1 \rightarrow 2 \rightarrow 2 \rightarrow 1$
   Output: true

   Input: $1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 1$
   Output: true

   Expected time: $O(n^2)$, Expected space: $O(1)$
   **Follow-Up:** Expected time: $O(n)$, Expected space: $O(n)$
   **Follow-Up 2:** Expected time: $O(n)$, Expected space: $O(1)$