**Data structures: Stacks and queues**

To succeed at this challenge, you'll need to demonstrate that you can do the following:

* Implement a stack and a queue.

**Instructions**

Your goal for this checkpoint is to get the tests to pass.

To do so, you will be modifying the existing Stack and Queue classes to implement a stack and queue with insertion and deletion capabilities.

**Existing files**

| **File path** | **Description** |
| --- | --- |
| src/Stack.js | Contains the definition of the Stack class. The constructor() method has already been completed for you. |
| src/Queue.js | Contains the definition of the Queue class. The constructor() method has already been completed for you. |

**Tasks**

Complete the following tasks to pass the tests and this assignment. Just like in the checkpoint, you will use a linked list in your implementation.

In the src/Stack.js file:

1. Complete the push() method to insert a node into the stack. The method should accept an argument data that represents the data to be stored in the node. Use the provided \_Node class to create new node instances.
2. Complete the pop() method to remove a node from a stack. The method should return the data stored in the deleted node.

In the src/Queue.js file:

1. Complete the enqueue() method to insert a node into a queue. The method should accept an argument data that represents the data to be stored in the node. Use the provided \_Node class to create new node instances.
2. Complete the dequeue() method to remove a node from a queue. The method should return the data stored in the deleted node. If the queue is already empty, then the method should return undefined.

Once these tasks are complete, all tests should pass.