MIE245 - Winter 2026
Lab 1 - Linked Lists

Deadline: Tuesday, January 13, 11:59pm EST

Submission Instructions

Reminder: You can not claim work that was completed by others or generative AI as your own. Submit a completed **readme** template file outlining what code you contributed and what code was completed by other students, people, tools, and/or leveraged from websites.

Add _**utorid** (with **utorid** replaced by your UTORid in all lower-case characters) to the end of both the completed template file names (i.e., the .py and the readme files). Please ensure you use your UTORid (i.e., what you use to sign into Quercus and **not** other identifiers such as your student number).

Compress your completed template files to a zip file and upload the zip file to Quercus.

Lab Instructions

In this lab, you will write a class implementation of a linked list and a node. A linked list is a collection of connected nodes. Each node has a value (in this lab, an integer) and a link to the next node (Figure 1). In addition, a linked list has a pointer to the start of the list (head) and a pointer to the end of the list (tail) (Figure 2).

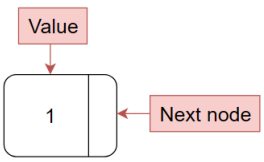

Figure 1: A node in a linked list

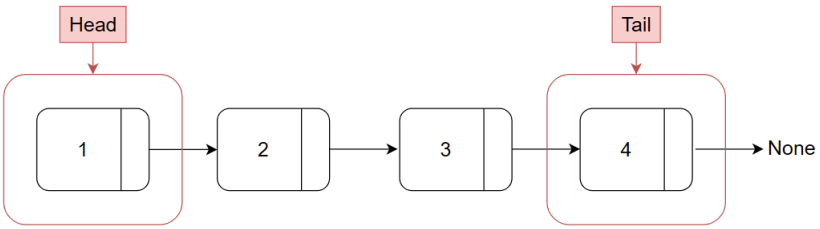

Figure 2: A linked list

For this lab, complete the template file **MIE245_Lab_1_Template.py** by writing the methods for the linked list class. The requirements for the methods are:

- *insert_start(self, value)*: insert a node at the start of the list
- *insert_end(self, value)*: insert a node at the end of the list
- *search_for_node_by_value(self, value)*: search for a node by its value. This function should return the node object. If the node does not exist, then return None.
- *delete_node_by_value(self, value)*: delete a node by its value. If the node does not exist, then the linked list is unchanged.

You may assume that any elements within the linked list are unique.

AutoGrading

Additionally, you have access to an example of the auto-grading script that will be used to test your submission (**MIE245_Lab_1_Autograder.py**). Please follow the steps below to run the auto-grader for the provided test case:

1. Place the auto-grader script in a new folder
2. Inside the newly created folder, create a folder called "submissions"
3. Place your script in the "submissions" folder
4. Run the auto-grader script and open the **grades.csv** file to see if you passed the test case (1 is pass, 0 is fail)

Note: The auto-grading script only contains one of the several test cases we will evaluate against, so even if you pass this test case, you may not have a completely correct solution.

Tip: You can modify the test case to test additional functionality.