

**Lab 2: ADT Stack Implementations**

The goal of this lab is to implement the Stack Abstract Data Type using two different implementations:

- 1) The built in List construct in Python (array)
- 2) The recursive node list data structure covered in class

For these two implementations, you will write the code for the basic operations of a stack.

In the case of the Python List implementation, you will allocate a list of size capacity and use this to store the items in the stack. Since Lists in Python expand when more storage is needed, you must avoid using any functions that would cause a List to expand (see the list of forbidden methods/operations below). This will allow you to treat the Python List as a simple array. For the recursive node list data structure implementation, we will not have a capacity attribute for the stack.

You are NOT allowed to use the following Python List operations (we will check!):

- `append()`
- `insert()`
- `extend()`
- `remove()`
- `pop()`
- `del()`
- `+` (concatenations)
- List slicing (except in the `__init__`, `__eq__` and `__repr__` functions provided to you in the starter code for the array implementation)

Additional Requirements:

- All stack operations must have  $O(1)$  performance (including the `size()` operation)
- Your stack implementations must be able to hold values of `None` as valid data

The following starter files are available on GitHub, and after you have added your code, these are the files you will commit and push for this lab:

- **`stack_array.py`**: Contains an array (Python List) based implementation of the **Stack** class
- **`stack_nodelist.py`**: Contains a linked based implementation of the **Stack** class
- **`stack_array_tests.py`**: Contains comprehensive tests to ensure your implementations in `stack_array.py` work correctly.
- **`stack_nodelist_tests.py`**: Contains comprehensive tests to ensure your implementations in `stack_linked.py` work correctly.

Note that the class in each stack implementation is named **Stack**, and both implementations follow the same specification in regard to the operations on the Stack, with the exception that the node list implementation does not include an `is_full()` function.