Assignment 3 **CSSE3100/7100 Reasoning about Programs**

Due: 4pm on 27 May, 2025

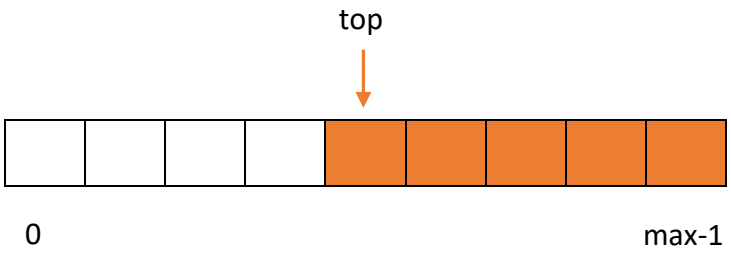
The aim of this assignment is to consolidate your understanding of the course's material on objects and array-based data structures. It is worth 15% of your final mark for the course.

Submission instructions: Upload a single Dafny file (A3.dfy) to Gradescope with your solutions to Q1-Q3. The file should be formatted as per the formatting instructions below.

IMPORTANT: If either of your file has syntax or other parsing errors, i.e., VS Code does not report "Verification Succeeded" or "Could not prove ..." then the autograder will not be able to run and your mark will be 0/15 marks.

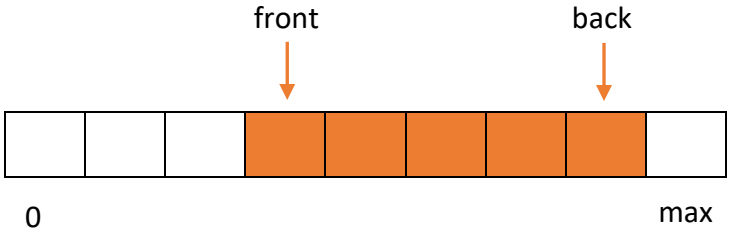
Code which does parse but reports "Could not prove ..." is acceptable and will receive part marks based on the marking criteria below.

1. Abstractly, a bounded stack can be represented by a sequence, s, and a natural number, max >= 0, representing its capacity (i.e., a stack with capacity 0 is allowed). The top of the stack (i.e., the last element added to it) is at s[0]. Concretely, this can be implemented using an array, a, of length max and a natural number, top, which keeps track of the position of the top of the stack as shown below (orange squares represent items in the stack).

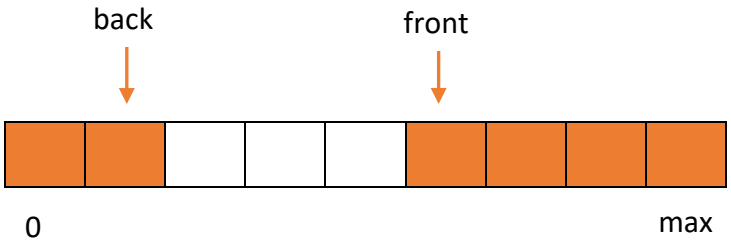


Following the approach from the lectures on objects and data structures, provide a specification and verified implementation of such a stack in Dafny. **(3 marks)**

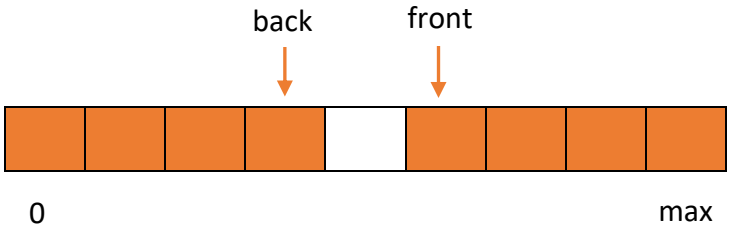
2. A bounded *deque* (double-ended queue) is a queue which allows items to be added and removed from either the front or back. Abstractly, it can be represented by a sequence, q, and a natural number, max >= 0, representing its capacity. The front of the deque is q[0]. Concretely, this can be implemented using an array, a, of length max+1 and two natural numbers, front and back, which keep track of the positions of the front and back of the deque as shown below.



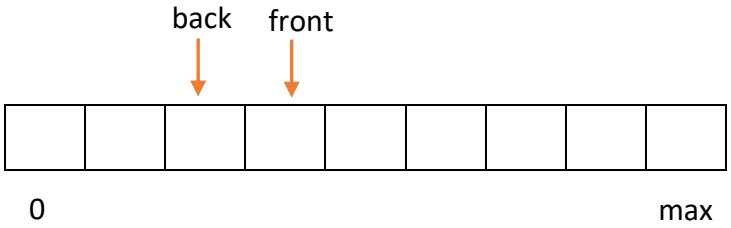
The array is *circular* allowing back to be before front in the array as follows.



A full deque is shown below. There is still one free space in the deque (since the array a has length max+1).



This additional space allows us to represent an empty deque as one where back == front - 1



or where front is 0 and back is max.

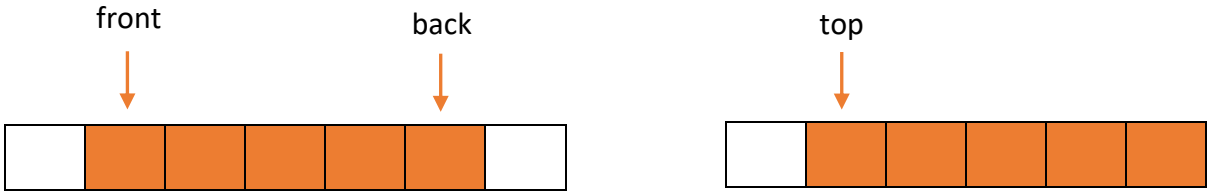
Following the approach from the lectures on objects and data structures, provide a specification and verified implementation of such a deque in Dafny. **(5 marks)**

3. We wish to implement a bounded stack which has an additional method PopMiddle (which returns and removes the middle element of a non-empty stack). Given the number of elements in the stack is n, the middle element of the stack is defined as that at position n/2 where / is integer division. For example, if the stack has 6 or 7 elements then the middle element is at position 3. If the stack has 1 element then the middle element is at position 0.

Abstractly the stack can be represented by a sequence, s, and a natural number, max >= 0, representing its capacity. The top of the stack is at s[0].

Concretely, the stack can be implemented using a simple stack (from Q1) and a deque (from Q2). To enable the middle element to be readily accessed, the first half of the stack will be stored in the deque and the second half in the simple stack. When there is an odd number of elements, the deque will have one more element than the stack.

Elements are pushed to and popped from the front of the deque, and transferred between the back of the deque and the simple stack to maintain the correct number of elements in each. A possible configuration of the stack is shown below.



In this configuration, there are 10 elements and hence the middle element is at position 5, which is the top element of the simple stack.

Following the approach from the lectures on objects and data structures, provide a specification and verified implementation of such a deque in Dafny. **(7 marks)**

Formatting instructions

The specification and code for Q1-Q3 should be added to the template file A3.dfy. To allow us to auto-grade your solution, do **not** add any state variables, methods, or functions. Do **not** change the types of variables nor the signatures (input, outputs and their types) of constructors, methods and functions.

Ensure that for Q1 and Q3, the top of the stack corresponds to s[0], and for Q2, the front of the deque corresponds to q[0].

Marking

You are expected to be able to correct syntax, parsing and similar errors which prevent the verifier being run. Failure to do so will result in a mark of 0.

For each question, you will be awarded full marks for code which has a correct abstraction relation and correct pre- and postconditions. You will also be awarded marks for code which verifies.

You will lose marks as detailed below.

Q1 You will lose 0.5 marks for not precisely following the approach from the lectures on objects and data structure. You will lose 0.5 marks for missing or additional conjuncts in the abstraction relation (capped at 1 marks). You will lose 0.5 marks for missing or additional conjuncts in the pre- and postconditions of methods (capped at 0.5 marks per method). You will lose 0.5 marks for missing or additional pre- and postconditions of the constructor or Size function (capped at 0.5 marks).

Q2 You will lose 0.5 marks for not precisely following the approach from the lectures on objects and data structure. You will lose 0.5 marks for missing or additional conjuncts in the abstraction relation (capped at 2 marks). You will lose 0.5 marks for missing or additional conjuncts in the pre- and postconditions of methods (capped at 0.5 marks per method). You will lose 0.5 marks for missing or additional pre- and postconditions of the constructor or Size function (capped at 0.5 marks).

Q3 You will lose 0.5 marks for not precisely following the approach from the lectures on objects and data structure. You will lose 0.5 marks for missing or additional conjuncts in the abstraction relation (capped at 3.5 marks). You will lose 0.5 marks for missing or additional conjuncts in the pre- and postconditions of methods (capped at 0.5 marks per method). You

will lose 0.5 marks for missing or additional pre- and postconditions of the constructor or Size function (capped at 0.5 marks).

You will lose 1 mark if your code does not verify.

School Policy on Student Misconduct

This assignment is to be completed individually. You are required to read and understand the School Statement on Misconduct, available on the School's website at: https://eecs.uq.edu.au/current-students/student-guidelines/student-conduct?p=1#1

This assessment task evaluates students' abilities, skills and knowledge without the aid of generative Artificial Intelligence (AI). Students are advised that the use of AI technologies to develop responses is strictly prohibited and may constitute student misconduct under the Student Code of Conduct.