CNIT 31500 – Lab 5

Purdue University

Fall 2023

November 5, 2023 version 1. Subsequent changes will be marked in blue.

1 Goals

The primary goal of this project is to use linked lists to build stack and queue data structures. The previous lab 4 tasked you with the construction of the single linked list with all of its structure and functions. This lab the use of the linked list to construct stacks and queues and compare them with array-based stacks and queues. The stacks and queues will then be used to manipulate data.

2 Submission Guidelines

Submit your code on Gradescope, NOT Brightspace. Do not indicate your name or username in the comments to allow for anonymous grading.

Your code must compile to get a non-0 credit.

2.1 Checkoff requirements

Stack code for both Linked List and Array must be checked off week of November 5.

3 Stack

Develop a stack data structure, using a linked list, with the following functions:

- PushLL - pushes a data node onto the top of the stack. This function will need a parameter of the data type stored in the stack.
- PopLL - Takes the top node of the stack and returns the data to the program.
- SizeLL - returns the size of the stack in terms of the number of nodes.
- EmptyLL - returns a boolean, true if the stack has no nodes and false if the stack has greater than 0 nodes.

Develop a stack data structure, using arrays, with the following functions:

- PushA - pushes a data node onto the top of the stack. This function will need a parameter of the data type stored in the stack.
- PopA - Takes the top node of the stack and returns the data to the program.
- SizeA - returns the size of the stack in terms of the number of nodes.
- EmptyA - returns a boolean, true if the stack has no nodes and false if the stack has greater than 0 nodes.

4 Queue

Develop a queue data structure using a linked list, with the following functions:

- EnqueueLL - adds a data node onto the back of the queue. This function will need a parameter of the data type stored in the queue.
- DequeueLL - Takes the front node from the queue and returns the data to the program.
- SizeLL - returns the size of the queue in terms of the number of nodes. (this can be the same function as in Stack)
- EmptyLL - returns a boolean, true if the queue has no nodes and false if the queue has greater than 0 nodes. (this can be the same function as in Stack)

Develop a queue data structure using an array, with the following functions:

- EnqueueA- adds a data node onto the back of the queue. This function will need a parameter of the data type stored in the queue.
- DequeueA- Takes the front node from the queue and returns the data to the program.
- SizeA- returns the size of the queue in terms of the number of nodes.
- EmptyA- returns a boolean, true if the queue has no nodes and false if the queue has greater than 0 nodes.

5 General Description

The description for each of the stack and queue functions were given in class. There must be a Node Struct containing data and a pointer. In this case, the data will be a Struct with the following data: First Name, Last Name, PUID and Age. You will need a start pointer to maintain the head of the list and a

current pointer when traversing the list. Hint - think about the order to write the functions, as some are dependent on others!

To use the stack and queue, you will develop a simple text-based interface. The interface will ask a user to input data for a Struct First Name, Last Name, PUID, Age. Once you get the 4 data elements, you add them to a Struct. Once they are added to a Struct, you will add the Struct to a stack and/or a queue, depending on a user request. Each user request should be executed for an array and a linked list.

During each iteration, the user will have the option to:

- Pop - delete a node from the Stack
- Push - add a node to the Stack
- Enqueue - add a node to the Queue
- Dequeue - delete a node from the queue
- Add element (both Push and Enqueue)
- Remove element (both Pop and Dequeue)
- Empty Queue - remove all of the nodes from the queue
- Empty Stack - remove all of the nodes from the stack
- Print Queue - print the nodes of the Queue in the order of arrival into the queue (both linked list and array)
- Print Stack - print the nodes of the Stack, which should be the reverse of the Queue (both linked list and array)
- Print Number of moves so far for Array and Linked List (separate number of the array, separate number for linked list)
- Exit - exit the program.

The number of moves should be calculated as how many times you need to move an element from position to a position (in case of array moves) or traverse a list (if needed); plus 1 move for the actual pop/push/enque/dequeue.