School of Science, Computing and Artificial Intelligence
The University of the West Indies, Five Islands

COMP0002 - Lab 2

Name: Ajante Fraser
ID: 500000947

School of Science, Computing and Artificial Intelligence
The University of the West Indies, Five Islands

Table of Contents

School of Science, Computing and Artificial Intelligence
The University of the West Indies, Five Islands

Exercise 1

1. Using an example, why is there a need for so many data structures?
There are many types of data structures because of the efficiency and specific use case of each. Based on the problem at hand programmers have access to various Data Structures that will be best suited for the problem so that it is efficient. For example, arrays are efficient for random access but may be slow for insertions and deletions, while linked lists are good for insertions and deletions but not as efficient for random access.

2. How does a circular queue differ from the standard queue?
A circular queue differs from a standard queue in that it has a fixed size and efficiently reuses available space by wrapping around when it reaches the end, allowing constant-time enqueue and dequeue operations without element shifting.

3. What are three benefits of class abstraction?

- Abstraction simplifies interaction with a program by hiding irrelevant data, reducing complexity.
- Abstraction lets programmers reuse pieces of their code in different places by promoting code reuse.
- Abstraction assists in testing and debugging. You can create mock implementations of abstract classes to facilitate testing and debug it if any issues arise.

Exercise 2

Operation	0	1	2	Front	Back
enqueue(40)	40			0	1
enqueue(10)	40	10		0	2
peek( )	40	10		0	2
enqueue(5)	40	10	5	0	3
dequeue( )		10	5	1	3
dequeue( )			5	2	3
enqueue(33)			5	2	3
dequeue( )				3	3
dequeue( )				3	3

Operation	0	1	2	Top
push(51)	51			0
pop( )				-1
push(-13)	-13			0
push(-6)	-13	-6		1
push(-12)	-13	-6	-12	2
pop( )	-13	-6		1
pop( )	-13			0
pop( )				-1
push(-12)	-12			0
push(40)	-12	40		1
peek( )	-12	40		1

Operation	0	1	2	Front	Back
enqueue(40)	40			0	1
enqueue(10)	40	10		0	2
peek( )	40	10		0	2
enqueue(5)	40	10	5	0	3
dequeue( )		10	5	1	3
dequeue( )			5	2	3
enqueue(33)	33		5	2	0
dequeue( )	33			0	1
dequeue( )				0	0

Exercise 3

LinkedList
+ head: **Node**
+ LinkedList( ) + add(item:**int**): **None** + delete( ): **int** + printList( ): **None**