ASSIGNMENT IN DSA

TUBTUB, ALLYSON MAE J. BSIS – II

**Pseudocode for implementing stacks using an array.**

The stack consists of an *N*-element array *S* and an integer variable *t*, the index of the top element in array *S*.

Array indices start at 0, so we initialize *t* to -1

**Algorithm** size():

return *t* +1

**Algorithm** isEmpty():

return (*t*<0)

**Algorithm** top():

**if** isEmpty() **then**

throw a StackEmptyException

return *S*[*t*]

**Algorithm** push(*o*):

**if** size() = *N* **then**

throw a StackFullException

*t* ¬ *t* + 1

*S*[*t*] ¬ *o*

**Algorithm** pop():

**if** isEmpty() **then**

throw a StackEmptyException

*e*¬*S*[*t*]

*S*[*t*]¬**null**

*t*¬*t*-1

**return** *e*

**Pseudocode for implementing Stacks using a SinglyLinkedList.**

1 public class LinkedStack<E> implements Stack<E> {

2 private SinglyLinkedList<E> list = new SinglyLinkedList<>( ); // an empty list

3 public LinkedStack( ) { } // new stack relies on the initially empty list

4 public int size( ) { return list.size( ); }

5 public boolean isEmpty( ) { return list.isEmpty( ); }

6 public void push(E element) { list.addFirst(element); }

7 public E top( ) { return list.first( ); }

8 public E pop( ) { return list.removeFirst( ); }

9 }

**Pseudocode for an Array-based implementation of a queue.**

**Algorithm** size():

**return** (*N - f + r*) mod *N*

**Algorithm** isEmpty():

**return** (*f = r*)

**Algorithm** front():

**if** isEmpty() **then**

throw a QueueEmptyException

**return** *Q*[*f*]

**Algorithm** dequeue():

**if** isEmpty() **then**

throw a QueueEmptyException

*temp* ¬ *Q*[*f*]

*Q*[*f*] ¬ **null**

*f* ¬ (*f* + 1) mod *N*

**return** *temp*

**Algorithm** enqueue(*o*):

**if** size = *N* - 1 **then**

throw a QueueFullException

*Q*[*r*] ¬ *o*

*r* ¬ (*r* +1) mod *N*

**Pseudocode for an Implementation of a Queue using a SinglyLinkedList.**

1 /\*\* Realization of a FIFO queue as an adaptation of a SinglyLinkedList. \*/

2 public class LinkedQueue<E> implements Queue<E> {

3 private SinglyLinkedList<E> list = new SinglyLinkedList<>( ); // an empty list

4 public LinkedQueue( ) { } // new queue relies on the initially empty list

5 public int size( ) { return list.size( ); }

6 public boolean isEmpty( ) { return list.isEmpty( ); }

7 public void enqueue(E element) { list.addLast(element); }

8 public E first( ) { return list.first( ); }

9 public E dequeue( ) { return list.removeFirst( ); }

10 }