CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

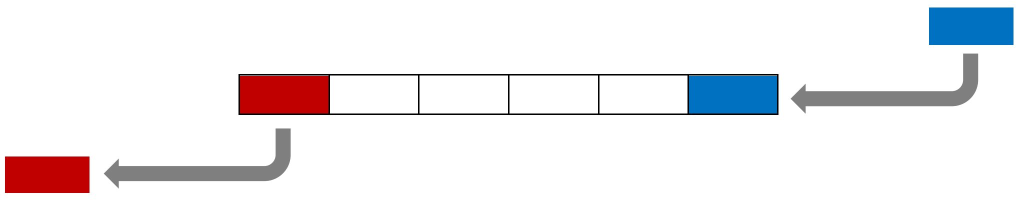
Computing@CIT

LINEAR DATA STRUCTURES AND ALGORITHMS.

ASSIGNMENT 1: DATA STRUCTURES - ADT MyQueue

BACKGROUND.

A queue is a linear data structure in which elements are to be accessed in the same order in which they were stored. It is also known as a First In First Out (FIFO) structure.



In this assignment we are going to specify the ADT MyQueue with the following operations:

- **createEmpty:** It creates a new MyQueue with no elements.
- **isEmpty:** It returns whether the queue is empty or not.
- **first:** If the queue is non-empty, then it returns its first element (coloured in red above). Otherwise, it returns an error message.
- **add:** Given a new item (coloured in blue above), it adds it at the back of the queue.
- **remove:** If the queue is non-empty, then it removes its first element (coloured in red above). Otherwise, it returns an error message.

ASSIGNMENT 1 – HINT 1 **(Week 4)**

A MyQueue of int elements: Static Implementation.

BACKGROUND.

The folder **/src** contains the following files:

- **MyMain.java:** This class tests the functionality of the queue's static implementation.
- **MyQueue.java:** This interface specifies the ADT MyQueue containing int elements.
- **MyStaticQueue.java:** This class implements all operations of MyQueue, using a static based implementation based on the following attributes:
    - private int items[];
    - private int numItems;
    - private int maxItems;

The folder **/doc** contains the documentation of the project. In particular:

- **MyMain.html:** Contains the description of the class MyMain.java.
- **MyQueue.html:** Contains the description of the interface MyQueue.java.
- **MyStaticQueue.html:** Contains the description of the class .java.

EXERCISE.

Implement the class MyStaticQueue.java.

ASSIGNMENT 1 – HINT 2 **(Week 5)**

A MyQueue of int elements: Dynamic Implementation.

BACKGROUND.

The folder **/src** contains the following files:

- **MyMain.java:** This class tests the functionality of the queue's dynamic implementation.
- **MyNode.java:** This class models the concept of a single linked node containing an int *info* and a pointer to its next node *next*.
- **MyQueue.java:** This interface specifies the ADT MyQueue containing int elements.
- **MyDynamicQueue.java:** This class implements all operations of MyQueue, using a dynamic based implementation based on the following attributes:
    - private MyNode head;

The folder **/doc** contains the documentation of the project. In particular:

- **MyMain.html:** Contains the description of the class MyMain.java.
- **MyNode.html:** Contains the description of the class MyNode.java.
- **MyQueue.html:** Contains the description of the interface MyQueue.java.
- **MyStaticQueue.html:** Contains the description of the class MyQueue.java.

EXERCISE.

Implement the class MyDynamicQueue.java.

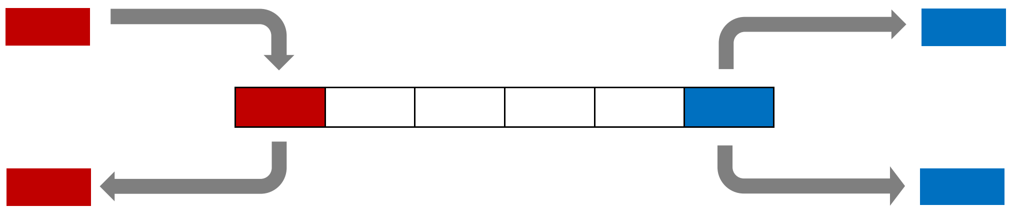
ASSIGNMENT 1 – HINT 3 **(Week 6)**

A MyQueue of generic type <T> elements: Dynamic Double-Linked Implementation.

BACKGROUND.

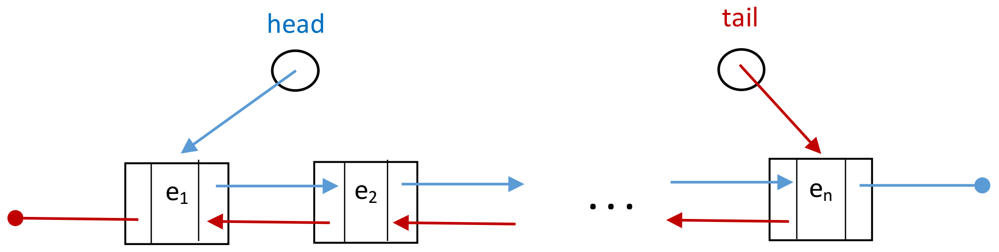
In this last hint we extend the ADT MyQueue with the capability of adding and removing elements from both front and back. The new extended ADT MyQueue contains the following operations:

- **createEmpty:** It creates a new MyQueue with no elements.
- **isEmpty:** It returns whether the queue is empty or not.
- **first:** If the queue is non-empty, then it returns its first element (coloured in red above). Otherwise, it returns an error message.
- **addByFirst:** Given a new item (coloured in red above), it adds it at the front of the queue.
- **removeByFirst:** If the queue is non-empty, then it removes its first element (coloured in red above). Otherwise, it returns an error message.
- **last:** If the queue is non-empty, then it returns its first element (coloured in blue above). Otherwise, it returns an error message.
- **addByLast:** Given a new item (coloured in blue above), it adds it at the back of the queue.
- **removeByFirst:** If the queue is non-empty, then it removes its last element (coloured in blue above). Otherwise, it returns an error message.



The folder **/src** contains the following files:

- **MyMain.java:** This class tests the functionality of the queue's dynamic implementation.
- **MyDoubleLinkedNode.java:** This class models the concept of a double linked node containing an element of type <T> *info*, a pointer to its next node *left* and a pointer to its previous node *right*.
- **MyQueue.java:** This interface specifies the ADT MyQueue containing elements of type <T>.
- **MyDoubleDynamicQueue.java:** This class implements all operations of MyQueue, using a dynamic based implementation based on the following attributes:
    - private MyDoubleLinkedNode<T> head;
    - private MyDoubleLinkedNode<T> tail;



The folder **/doc** contains the documentation of the project. In particular:

- **MyMain.html:** Contains the description of the class MyMain.java.
- **MyDoubleLinkedNode.html:** Contains the description of the class MyNode.java.
- **MyQueue.html:** Contains the description of the interface MyQueue.java.
- **MyDoubleDynamicQueue.html:** Contains the description of the class MyQueue.java.

EXERCISE.

Implement the class MyDoubleDynamicQueue.java.

MARK BREAKDOWN.

Assignment 1: 25 marks.

- Hint 1: 7.5 marks

public MyStaticQueue(int m){...}	1.5 marks
public boolean isEmpty( ){...}	1.5 marks
public int first( ){...}	1.5 marks
public void add(int element){...}	1.5 marks
public void remove( ){...}	1.5 marks

- Hint 2 → 7.5 marks.

public MyDynamicQueue( ){...}	1.5 marks
public boolean isEmpty( ){...}	1.5 marks
public int first( ){...}	1.5 marks
public void add(int element){...}	1.5 marks
public void remove( ){...}	1.5 marks

- Hint 3 → 10 marks.

public MyDoubleDynamicQueue( ){...}	1 marks
public boolean isEmpty( ){...}	1 marks
public int first( ){...}	1 marks
public void addByFirst(int element){...}	1.5 marks
public void removeByFirst( ){...}	1.5 marks
public int last( ){...}	1 marks
public void addByLast(int element){...}	1.5 marks
public void removeByLast( ){...}	1.5 marks

To evaluate each function I will run it over some tests. The results are based on the performance of your code over these tests.

For each function, there are 3 possible scenarios:

A. The function passes all the test → 100% of marks.
B. The function does not pass all tests by small details → 100% of marks – 0.25/0.5 marks (depending how small is the mistake).
C. The function does not pass all tests by big details, it does not compile or it generates and exception (makes the program crash) or the function was not attempted → 0% of marks.

IMPORTANT: I will use a source code plagiarism detection tool. In case of detecting a copy (for at least one method) between some students, ALL these students get 0% marks for the WHOLE assignment. INCLUDING the student that originally coded it.

SUBMISSION DETAILS.

Deadline.

Sunday 22nd of October, 11:59pm.

Submission Details.

Please submit to Blackboard (folder Submissions, A01) **ONLY** the following files:

- Hint 1 → File “MyStaticQueue.java”.
- Hint 2 → File “MyDynamicQueue.java”.
- Hint 3 → File “MyDynamicDoubleQueue.java”.

Lab Demo.

A brief individual interview about the assignment will take place on our lab session on week 7 (Monday 23rd - Friday 27th of October).
The demo is mandatory for the assignment to be evaluated.

Please, let me know in the lab if you are willing to do the **demo earlier** than week 7. (With the corresponding uploading of the full assignment to the blackboard). However, I warn you that once the demo is done, there will be **no possibility of re-evaluating later.** The evaluation will be done with the exact code presented at this moment. Therefore, my **advice** is to take this option only if you are **100% sure** that your assignment is already perfect.