Lab 3: Queue data structure

[deadline**: 22th April 2022]**

Prologue

A *queue* is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence.

The operations of a queue make it a first-in-first-out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed.

Task

Define the class `queue` to be a FIFO data structure – the element that was added to this structure earliest will be pulled from it the fastest. This structure is to be used to store strings.

The functionality of the queue should be very simple: put the string into the queue (method `void enqueue(string)`), take the string from the queue (method `string dequeue()`), check what the string is at the beginning of the queue (method `string front()`) and ask for the number of all items in the queue (method `int size()`).

Design this class using one dimensional array allocated on the heap by operator `new[]`. The destructor must free the memory allocated to this array by operator `delete[]`. Implementing the queue, use the wrapping in the array – the first element of the array follows the last one.

The capacity of the queue should be specified in the constructor – define a private field `capacity` to remember the maximum size of the queue. You will also need information about the place where the queue begins and the number of items currently stored in the queue – define a private fields `start` and `length` to remember this parameters.

The class `queue` should have five constructors: a constructor with a given capacity, a default constructor (without any arguments), a constructor initializing the queue with a list of strings (using `initializer_list<string>`), a copy constructor and move constructor. To complete the copy and move semantics define the appropriate assignment operators (copy and move assignments).

Finally, write an interactive program for testing queue (interpret and execute commands from console). The queue object that you will test create on the heap with operator `new` and remove it with operator `delete` before the program ends!

Whenever we encounter any errors, ambiguities or contradictions in the program, this should be signaled by an exception.