Exercises about Queues and Deques

Exercise 1

There are two main possibilities for the implementation of queues: arrays and singly linked lists. Based on that, implement queues using each one of the following underlaying structures:

- arrays
- growable arrays
- singly linked lists

Exercise 2)

Create a program that tests the spatial and runtime efficiency of the programs implemented for the previous exercise (Exercise 1). This program might, for instance, create several queues of varying sizes, using each one of the previous queue implementations. After that, the program can select a sequence of operations to be applied over each queue. During the execution of these operations the program collects information regarding execution time and consumed space. The obtained results can be plotted for comparison. Try to explain the differences among the obtained results.

Exercise 3)

Considering the following deque operations:

- **push_left**: inserts a new element at the left of the deque.
- **push_right**: inserts a new element at the right of the deque.
- **pop_left**: removes a element from the left of the deque.
- **pop_right**; removes a element from the left of the deque.
- **first**: returns the leftmost element of the deque without actually removing it.
- **last**: returns the rightmost element of the deque without actually removing it.

Write down a table presenting the complexities of the above operations considering that the deque was implemented with an (sufficiently large) array, a singly linked list and a doubly.

Exercise 4)

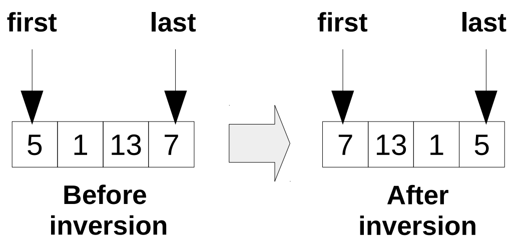
Implement a deque using a circular growable array.

Exercise 5)

Write a program that uses the deque implemented in the previous exercise to verify if a word is palindrome.

Exercise 6)

Write a function, in C, that inverts the order of the elements of a queue. Example:



The above mentioned queue is implemented through the following class:

```
class Queue
{
public:
      ...
      void Push(...);
      int Pop();

      int array[10];
      int first;
      int last;
}
```

Your function must necessarily use a stack as an auxiliary data structure to invert the queue!
The function that you have to write must be declared as:

```
void invertQueue(class *Queue);
```