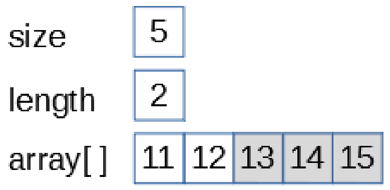
# WORKSHEET, WEEK 04: STACKS AND QUEUES

**Question 1.1.1 (Stack Implementation as Array):**

Stack is implemented as an array. In our case the array has size $n = 5$. Stack contains integer numbers; initially the array has the following content.



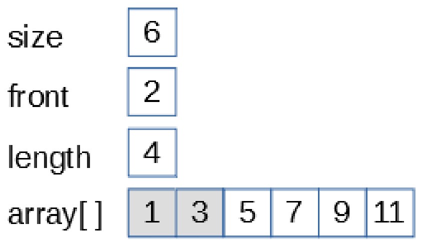
Stack has the physical representation with `length` $= 2$ (the number of elements in the stack), `size` $= 5$ (maximal number of elements contained in the stack). We have the following fragment:

```
pop();
push(21);
push(22);
pop();
push(23);
push(24);
pop();
push(25);
```

Draw the state of the array after every command. (Every `push(elt)` command assigns a new element into the element `array[length]`, then increments `length` by 1. The command `pop()` does not modify the array, but decreases `length` by 1.

If the command cannot be executed (`pop()` on an empty stack; `push(elt)` on a full stack), then the stack structure does not change at all (`array` or `length` are not modified). To help imagine the state of this stack, you can shade those cells that do not belong to the array.

**Question 1.1.2 (Queue Implementation as a Circular Array):** A queue is implemented as an array with `size` elements; it has two extra variables `front` (pointer to the first element) and `length` (the current number of elements in the queue). Current state is shown in the figure:



Enumeration of array elements starts with $0$. The array is filled in a circular fashion. The command `enqueue(elt)` inserts a new element at

$$(\texttt{front} + \texttt{length}) \bmod \texttt{size},$$

where "mod" means the remainder when dividing by `size`. It also increments the `length` element.

The command `dequeue()` does not change anything in the array, but increments `front` by $1$ and decreases $length$ by $1$. Thus the queue becomes shorter by $1$.

```
dequeue();
enqueue(21);
dequeue();
enqueue(22);
enqueue(23);
enqueue(24);
dequeue();
```

Show the state of the array after every command – `array, length, front` variables after every line. (Shade the unused cells.)

**Question 1.1.3:** Denote $a, b, c$ to be the last 3 digits of your Student ID, and compute the following numbers:

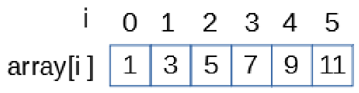
- $F = ((a + b + c) \mod 3) + 2$
- $\texttt{x1} = (a + b + c) \mod 10$
- $\texttt{x2} = ((a + b) \cdot 2) \mod 10$
- $\texttt{x3} = ((b + c) \cdot 3) \mod 10$
- $\texttt{x4} = ((c + a) \cdot 7) \mod 10$

The queue $Q$ is implemented as an array of size $N = 6$; its elements have indices from $\{0, 1, 2, 3, 4, 5\}$.

Initially the queue parameters are these:

- $\texttt{Q.front} = \texttt{F}$,
- $\texttt{Q.length} = 4$,
- $\texttt{Q.size} = 6$,

And the content of the array is the following:



Somebody runs the following code on this queue:

```
Q.enqueue(x1)
Q.enqueue(x2)
Q.dequeue()
Q.dequeue()
// show the state of Q
Q.enqueue(x3)
Q.enqueue(x4)
Q.dequeue()
// show the state of Q
```

After Line 4 (and at the very end) show the current state of the queue `Q`. The state should display the content of the array and also the values of `Q.front` and `Q.length`.

You can use shading, if it helps to visualize the array cells that are not currently used by your queue.

---

**Note:** Painting something gray is not required (since front/length indicate the state of your queue anyway). But painting cells gray may be helpful, if you want to visualize where your queue has the useful values (and what is some old garbage – you can shade it over).

---