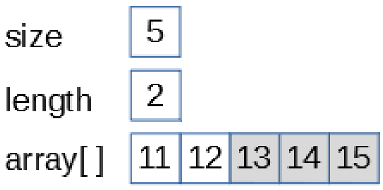
Sample Problems: Stacks and Queues

Published on 2020-10-19

Warm-up exercises offered before the midterm.

(1) Stack Implementation.

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.



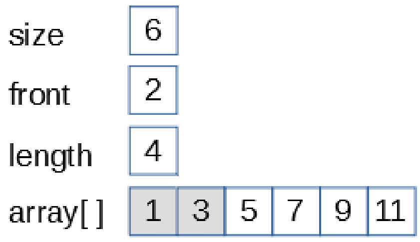
Stack has the physical representation with `length` = 2 (the number of elements in the stack), `size` = 5 (maximal number of elements contained in the stack). We have the following fragment:

```
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 (either `array` or `length`). To help imagine the state of this stack, you can shade those cells that do not belong to the array.

(2) Queue Implementation.

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, it 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.)

Note. In the actual midterm the starting state may be different, other command sequence (may also include conditionals and/or loops), it may be parametrized by the digits of your Student ID or by numbers computed from these digits.

Solutions

(1) Stack Implementation:

```
1 pop();
2 push(21);
3 push(22);
4 pop();
5 push(23);
6 push(24);
7 pop();
8 push(25);
```

	array[ ]					length
at start	11	12	13	14	15	2
after line 1	11	12	13	14	15	1
after line 2	11	21	13	14	15	2
after line 3	11	21	22	14	15	3
after line 4	11	21	22	14	15	2
after line 5	11	21	23	14	15	3
after line 6	11	21	23	24	15	4
after line 7	11	21	23	24	15	3
after line 8	11	21	23	25	15	4

(2) Queue Implementation:

```
1 dequeue();
2 enqueue(21);
3 dequeue();
4 enqueue(22);
5 enqueue(23);
6 enqueue(24);
7 dequeue();
```

	array[ ]						front	length
at start	1	3	5	7	9	11	2	4
after line 1	1	3	5	7	9	11	3	3
after line 2	21	3	5	7	9	11	3	4
after line 3	21	3	5	7	9	11	4	3
after line 4	21	22	5	7	9	11	4	4
after line 5	21	22	23	7	9	11	4	5
after line 6	21	22	23	24	9	11	4	6
after line 7	21	22	23	24	9	11	5	5