# Quiz 4

## (solutions)

1. (30 pts.) Given a circular array-based queue Q capable of holding 7 objects. Show the final contents of the array after the following code is executed:

```
for (int k = 1; k <= 7; k++)
  Q.enqueue(k);
for (int k = 1; k <= 4; k++)
{
  Q.enqueue(Q.dequeue());
  Q.dequeue();
}
```

Final answer:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | **3** | **5** | **7** | | | |

2. (20 pts.) Given a 5 element queue Q (from front to back: 1, 3, 5, 7, 9), and an empty stack S, remove the elements one-by-one from Q and insert them into S, then remove them one-by-one from S and re-insert them into Q. The queue now looks like (from front to back)

**9, 7, 5, 3, 1**

3. (10 pts.) What is the reason for using a "circular queue" instead of a regular one?

   **a. reuse empty spaces**

   b. running time of enqueue() is improved

   c. you can traverse all the elements more efficiently

   d. none of the above

4. (10 pts.) Assume that you have an empty circular queue (array-based implementation) which can hold only four elements. What are the array indexes of the back and the front elements after executing this series of queue operations? (indexes start with 0)

```
enqueue("a"), enqueue("b"), getFront(), enqueue("c"), enqueue("d"),
dequeue(), dequeue(), dequeue(), enqueue("a"), enqueue("b"),
enqueue("c"), dequeue(), getFront(), enqueue("d"), dequeue()
```

   front index _____**1**_____ back index _____**3**_____

5. (10 pts.) How would you access elements of an aggregated object (think of a collection) sequentially without exposing the underlying structure of the object?

   a) using indexes

   **b) using an iterator**

   c) using a stack.

   d) using a queue

6. (10 pts.) What is an interface?

   a) An interface is a collection of public methods of a class.

   b) **An interface is a collection of constants and method declarations.**

   c) An interface is a collection of methods and method declarations.

   d) none of the above

7. (10 pts.) The stack data type is restrictive in a sense that you cannot

   a) remove the top element

   b) insert at the top

   c) look at the top element

   **d) remove the bottom element**