**1** Add a variable to our `Queue` class in order to allow the `size()` method to be run in $O(1)$ (i.e., constant) time. Make all appropriate changes to our implementation of `Queue` to utilize this new variable. (1)

**2** Explain how you can use two stacks to emulate a queue. (1)

**3** Suppose that a client performs an intermixed sequence of `enqueue()` and `dequeue()` operations on a queue of integers. The enqueue operations put the integers $0$ through $9$, in order, onto the queue; the dequeue operations print out the return value. Which of the following sequence(s) could not occur? (2)

a. 0 1 2 3 4 5 6 7 8 9
b. 4 6 8 7 5 3 2 9 0 1
c. 2 5 6 7 4 8 9 3 1 0
d. 4 3 2 1 0 5 6 7 8 9

**4** A *priority queue* is an abstract data structure which adds a *priority* attribute to each element within it. Elements with a high priority "jump the queue" and are served first. Describe an application for which a priority queue would be more useful than a traditional queue. (2)

**5** *The Josephus Problem.* In the Josephus problem from antiquity, $N$ people are in dire straits and agree to the following strategy in order to reduce their population to a single person. They arrange themselves in a circle (at positions numbered $0$ through $N-1$) and proceed around the circle, eliminating every $M$th person until only one person is left. Legend has it that Josephus figured out where to sit in order to avoid being eliminated. Write a method, `Josephus()`, that takes $N$ and $M$ as parameters and uses `Queue` to print out the order in which people are eliminated (and thus would show where Josephus should sit in the circle). (4)

**Example:** `Josephus(7, 2)` would print: `1 3 5 0 4 2 6`.

**6** *Circular Queue.* Due to its nature, a queue that uses an array for its data storage could simultaneously be full and empty. In other words, it may have neither any elements in it nor any room to hold additional elements. (6)

(a) Modify our `Queue` implementation to make it *circular*. That is, if there is space at the beginning of the array, but none at the end, the data should "wrap around" to use the available space.

(b) Briefly explain why this approach is *more efficient* than shifting the elements of the queue forward after each `dequeue` operation.