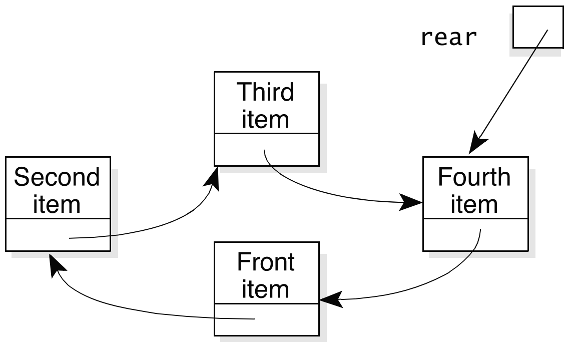
more passages until the user wishes to quit the program. Note that to test your program, you need not use such well- constructed poems. Your program will check any passage, regardless of its literary merit.

3 Enhance the car wash simulation method in Figure 7.8 on page 381 so that it has the following additional property. There is an additional parameter, which is a maximum length for the queue. When the queue gets as long as this maximum, any customer who arrives will leave without entering the queue (because the customer does not want to wait that long). There should also be one additional simulation result that is printed. In addition to the output shown in Figure 7.8, the method should print the number of simulated customers who left because the queue was too long. Embed the method in a program that allows the user to repeat simulations with different arguments until the user wishes to quit the program.

4 Give a complete implementation of a priority queue using an array of ordinary queues. For your ordinary queue, use the version from `edu.colorado.collections.ArrayQueue` in Figure 7.9 on page 387.

5 Give a complete implementation of a priority queue using the idea from the direct implementation on page 403.

6 In this chapter, we gave a linked list implementation of a queue. This implementation used two references, called `front` and `rear`, to refer to the front and the rear nodes of the queue (linked list). A **circular linked list** is similar to a regular linked list, except that the link field in the "last node" refers back to the "first node." (Of course, after this change, it is no longer clear which node, if any, is intrinsically "first.") If we use a circular linked list, then we need only one reference to implement a queue since the front node and the rear node are adjacent nodes, as at the top of the next column.



In the diagram, we have called the single reference `rear` because it refers to the last node in the queue. It turns out that this gives a more efficient implementation than having it refer to the first node in the queue. Redo the queue class using a circular linked list.

7 A **double-ended queue** is a list that allows the addition and removal of items from either end. One end is arbitrarily called the **front** and the other the **rear**, but the two ends behave identically. Specify, design, and implement a class for a double-ended queue. Include operations to check if it is empty and to return the number of items in the list. For each end, include operations for adding and deleting items. Implement the double-ended queue as a doubly linked list. Call your class `Deque` (pronounced "deck").

8 Make improvements to the car wash simulation program from Section 7.2. One particular improvement you should make is to handle the customers who are still in the queue at the end of the simulation. These customers should have their cars washed one after another, but no new customers should be allowed to join the queue during this time. The wait times of these leftover customers should be counted along with all the other customers.

9 Write a simulation program for a small airport that has only one runway. There will be a queue of planes waiting to land and a queue of planes waiting to take off. However, only one plane can use the runway at a time, so there can be only one takeoff or one landing in progress at any one time. Assume that all takeoffs take the same amount of time. Assume that all landings take the same amount of time, but this need not be the same as the takeoff time. Assume that planes arrive for landing at random times but with a specified probability of a plane arriving during any given minute. Similarly, assume that planes arrive at the takeoff queue at random times but with a (possibly different) specified probability of a departure. (Despite the fact that takeoffs and landings are scheduled, delays make this a reasonable assumption.) Since it is more expensive and more dangerous to keep a plane waiting to land than it is to keep a plane waiting to take off, landings will have priority over takeoffs. Thus, as long as some plane is waiting to land, no plane can take off. Use a clock that is an integer variable that counts the number of minutes simulated. Use a random number generator to simulate arrival and departure times of airplanes.

This simulation can be used, among other things, for deciding when the air traffic has become so heavy that a second runway must be built. Hence, the simulation will simulate disaster conditions in which planes crash because they run out of fuel while waiting too long in the landing queue. By examining the simulated situation, the airport authority hopes to avoid real tragedy. Assume all planes can remain in the queue for the same amount of time before they run out of fuel. If a plane runs out of fuel, your simulation will not discover this until the simulated plane is removed from the queue; at that point, the fact that the plane crashed is recorded, that plane is discarded, and the next plane is processed. A crashed plane is not considered in the calculation of waiting time. At the end of the simulated time, the landing queue is examined to see whether any of the planes in the simulated queue have crashed. You can disregard the planes left in the queue at the end of the simulation, other than those that crashed for lack of sufficient fuel. Use the following input and output specifications:

Input: (1) The amount of time needed for one plane to land; (2) the amount of time needed for one plane to take off; (3) the average amount of time between arrival of planes to the landing queue; (4) the average amount of time between arrival of planes to the takeoff queue; (5) the maximum amount of time that a plane can stay in the landing queue without running out of fuel and crashing; and (6) the total length of time to be simulated.

Output: (1) The number of planes that took off in the simulated time; (2) the number of planes that landed in the simulated time; (3) the number of planes that crashed because they ran out of fuel before they could land; (4) the average time that a plane spent in the takeoff queue; and (5) the average time that a plane spent in the landing queue.

10 Do an airplane simulation that is more complex than the previous project. In this version, planes arrive for landing with a random amount of fuel, which determines how long they can remain in the air. A plane will crash unless it lands within the time assigned to it. Your simulation will keep planes in a priority queue, in which the priority of a plane is equal to the number of minutes before midnight that the plane will crash.

11 Write a simulation program of the lines at a grocery store. The program will be similar to the car wash simulation, except that there are multiple queues instead of one. You might use an array (or vector) of queues to simulate the lines. Assume that there are five cashier lines at the grocery store. Customers enter randomly to check out, and then enter the shortest line. If the lines are equal, then the first available line is chosen. Each transaction takes a random amount of time to complete.

For additional work, expand the grocery line program to allow shoppers to:

- avoid a line if all lines are a certain length
- leave a line if they have waited beyond a certain time
- check if another line is shorter at specified time intervals
- switch lines if another line is shorter

12 Write a program that uses a priority queue to store a list of prioritized chores.

13 Use a circular array or a doubly linked list to implement a deque.