To simulate multiple servers, you will need to provide a separate customer status structure for each server. In the processing loop, each server should be tested for completion. Similarly, in start new call, a call should be started for each idle server.

You will also need to modify the call simulator to handle up to three customers arriving at the same time. To determine the number of customers, generate a random number in the range of 0 to 3. If the number is 0, no customer arrived. If the number is not 0, enqueue the number of customers indicated by the random number.

**28.** Write a stack and queue test driver. A test driver is a program created to test functions that are to be placed in a library. Its primary purpose is to completely test functions; therefore, it has no application use.

The functions to be tested are create stack, create queue, push stack, pop stack, enqueue, and dequeue. You may include other stack and queue functions as required. All data should be integers. You will need two stacks and two queues in the program, as described below.

**a.** Input stack: used to store all user input
**b.** Input queue: used to store all user input
**c.** Output stack: used to store data deleted from input queue
**d.** Output queue: used to store data deleted from input stack

Use a menu-driven user interface that prompts the user to select either insert or delete. If an insert is requested, the system should prompt the user for the integer to be inserted. The data are then inserted into the input stack and input queue. If a delete is requested, the data are deleted from both structures: The data popped from the input stack are enqueued in the output queue, and the data dequeued from the input queue are pushed into the output stack.

Processing continues until the input structures are empty. At this point, print the contents of the output stack while deleting all of its data. Label this output "Output Stack." Then, print all of the data in the output queue while deleting all of its data. Label this output "Output Queue." Your output should be formatted as shown below.

```
Output Stack: 18  9 13  7  5  1
Output Queue:  7 13  9 18  5  1
```

Test your program with the following operations:

```
1 input 1    5 delete     9 input 6    13 input 8
2 input 2    6 input 0   10 delete     14 delete
3 delete     7 input 5   11 input 7    15 delete
4 input 3    8 delete    12 delete     16 delete
```

In addition to the computer output from your test, write a short report (less than one page) describing what structural concepts were demonstrated by your output.

**29.** Queues are commonly used in network systems. For example, e-mail is placed in queues while it is waiting to be sent and after it arrives at the

recipient's mailbox. A problem occurs, however, if the outgoing mail processor cannot send one or more of the messages in the queue. For example, a message might not be sent because the recipient's system is not available.

Write an e-mail simulator that processes mail at an average of 40 messages per minute. As messages are received, they are placed in a queue. For the simulation, assume that the messages arrive at an average rate of 30 messages per minute. Remember, the messages must arrive randomly, so you will need to use a random number generator to determine when messages are received (see "Queue Simulation," on page 231).

Each minute, you can dequeue up to 40 messages and send them. Assume that 25% of the messages in the queue cannot be sent in any processing cycle. Again, you will need to use a random number to determine whether a given message can be sent. If it can't be sent, put it back at the end of the queue (enqueue it).

Run the simulation for 24 hours, tracking the number of times each message had to be requeued. At the end of the simulation, print the statistics that show

**a.** The total messages processed
**b.** The average arrival rate
**c.** The average number of messages sent per minute
**d.** The average number of messages in the queue in a minute
**e.** The number of messages sent on the first attempt, the number sent on the second attempt, and so forth
**f.** The average number of times messages had to be requeued (do not include the messages sent the first time in this average)