number of servers busy, divided by the number of servers. Note that this will be a number between 0 and 1.

(*d*) Now suppose that the bank's lobby is large enough to hold only 25 customers in the queues (total). If a customer arrives to find that there are already a total of 25 customers in the queues, he or she just goes away and the business is lost; this is called *balking* and is clearly unfortunate. Change the program to reflect balking, where the capacity of 25 should be read in as an input parameter. In addition to all the other output measures, observe the number of customers who balk during the course of the simulation.

2.5. In the manufacturing-system model of Sec. 2.7, correct the minor error described in the report generator regarding the collection of the total job delay in queue by job type. To do this, add an attribute to each job representing the cumulative delay in queue so far. When the job leaves the system, tally this value in sampst. Rerun the simulation for this alternative approach using the "current configuration" of the number of machines at each station.

2.6. For the manufacturing-system model of Sec. 2.7, estimate the expected overall average job time in system, being the weighted average of the expected times in system (delays in queue plus processing times) for the three job types, using the probabilities of occurrence of the job types as the weights. (*Hint*: You won't need a computer to do this.)

2.7. For the original configuration of the manufacturing system of Sec. 2.7, run the model for 100 eight-hour days, but use only the data from the last 90 days to estimate the quantities of interest. In effect, the state of the system at time 10 days represents the initial conditions for the simulation. The idea of "warming up" the model before beginning data collection is a common simulation practice, discussed in Sec. 9.5.1. (You may want to look at the code for simlib routine timest in Fig. 2.57, paying special attention to the variable treset, to understand how the continuous-time statistics will be computed.)

2.8. For the manufacturing-system model of Sec. 2.7, suggest a different definition of the attributes that would simplify the model's coding.

2.9. Do Prob. 1.15, except use simlib. Use stream 1 for interarrival times, stream 2 for service times at server 1, stream 3 for service times at server 2, and stream 4 for the travel times.

2.10. Do Prob. 1.22, except use simlib. Use stream 1 for the machine-up times and stream 2 for the repair times.

2.11. Do Prob. 1.24, except use simlib. Use stream 1 for interarrival times and stream 2 for service times. Note how much easier this model is to simulate with the list-processing tools.

2.12. Do Prob. 1.26, except use simlib. Use stream 1 for interarrival times, stream 2 for determining the customer type, stream 3 for service times of type 1 customers, and stream 4 for service times of type 2 customers.

2.13. Do Prob. 1.27, except use simlib. Use streams 1 and 2 for interarrival times and service times, respectively, for regular customers, and streams 3 and 4 for interarrival times and service times, respectively, of express customers.

2.14. Do Prob. 1.28, except use simlib. Use stream 1 for interarrival times for regular cars and stream 2 for service times for all cars.

2.15. Do Prob. 1.30, except use simlib. Use stream 1 for interarrival times, stream 2 for inspection times, stream 3 to decide whether a bus needs repair, and stream 4 for repair times.

2.16. For the inventory example of Sec. 1.5, suppose that the delivery lag is distributed uniformly between 1 and 3 months, so there could be between 0 and 3 outstanding orders at a time. Thus, the company bases its ordering decision at the beginning of each month on the sum of the (net) inventory level [denoted by $I(t)$ in Sec. 1.5] and the inventory on order; this sum could be positive, zero, or negative. For each of the nine inventory policies, run the model for 120 months and estimate the expected average total cost per month and the expected proportion of time there is a backlog. Note that holding and shortage costs are still based on the net inventory level. Use stream 1 for interdemand times, stream 2 for demand sizes, and stream 3 for delivery lags.

2.17. Problem 1.18 described a modification of the inventory system of Sec. 1.5 in which the items were perishable. Do this problem, using simlib, and in addition consider the case of LIFO (as well as FIFO) processing of the items in inventory. Use the same stream assignments as in Prob. 2.16, and in addition use stream 4 for the shelf lives.

2.18. For the time-shared computer model of Sec. 2.5, suppose that instead of processing jobs in the queue in a round-robin manner, the CPU chooses the job from the queue that has made the fewest number of previous passes through the CPU. In case of ties, the rule is FIFO. (This is equivalent to using the time of arrival to the queue to break ties.) Run the model with $n = 60$ terminals for 1000 job completions.

2.19. Ships arrive at a harbor with interarrival times that are IID exponential random variables with a mean of 1.25 days. The harbor has a dock with two berths and two cranes for unloading the ships; ships arriving when both berths are occupied join a FIFO queue. The time for one crane to unload a ship is distributed uniformly between 0.5 and 1.5 days. If only one ship is in the harbor, both cranes unload the ship and the (remaining) unloading time is cut in half. When two ships are in the harbor, one crane works on each ship. If both cranes are unloading one ship when a second ship arrives, one of the cranes immediately begins serving the second ship and the remaining service time of the first ship is doubled. Assuming that no ships are in the harbor at time 0, run the simulation for 90 days and compute the minimum, maximum, and average time that ships are in the harbor (which includes their time in berth). Also estimate the expected utilization of each berth and of the cranes. Use stream 1 for the interarrival times and stream 2 for the unloading times. [This problem is a paraphrasing of an example in Russell (1976, p. 134).]

2.20. Jobs arrive at a single-CPU computer facility with interarrival times that are IID exponential random variables with mean 1 minute. Each job specifies upon its arrival the maximum amount of processing time it requires, and the maximum times for successive jobs are IID exponential random variables with mean 1.1 minutes. However, if m is the specified maximum processing time for a particular job, the actual processing time is distributed uniformly between $0.55m$ and $1.05m$. The CPU will never process a job for more than its specified maximum; a job whose required processing time exceeds its specified maximum leaves the facility without completing service.

number of servers busy, divided by the number of servers. Note that this will be a number between 0 and 1.

(*d*) Now suppose that the bank's lobby is large enough to hold only 25 customers in the queues (total). If a customer arrives to find that there are already a total of 25 customers in the queues, he or she just goes away and the business is lost; this is called *balking* and is clearly unfortunate. Change the program to reflect balking, where the capacity of 25 should be read in as an input parameter. In addition to all the other output measures, observe the number of customers who balk during the course of the simulation.

2.5. In the manufacturing-system model of Sec. 2.7, correct the minor error described in the report generator regarding the collection of the total job delay in queue by job type. To do this, add an attribute to each job representing the cumulative delay in queue so far. When the job leaves the system, tally this value in sampst. Rerun the simulation for this alternative approach using the "current configuration" of the number of machines at each station.

2.6. For the manufacturing-system model of Sec. 2.7, estimate the expected overall average job time in system, being the weighted average of the expected times in system (delays in queue plus processing times) for the three job types, using the probabilities of occurrence of the job types as the weights. (*Hint*: You won't need a computer to do this.)

2.7. For the original configuration of the manufacturing system of Sec. 2.7, run the model for 100 eight-hour days, but use only the data from the last 90 days to estimate the quantities of interest. In effect, the state of the system at time 10 days represents the initial conditions for the simulation. The idea of "warming up" the model before beginning data collection is a common simulation practice, discussed in Sec. 9.5.1. (You may want to look at the code for simlib routine timest in Fig. 2.57, paying special attention to the variable treset, to understand how the continuous-time statistics will be computed.)

2.8. For the manufacturing-system model of Sec. 2.7, suggest a different definition of the attributes that would simplify the model's coding.

2.9. Do Prob. 1.15, except use simlib. Use stream 1 for interarrival times, stream 2 for service times at server 1, stream 3 for service times at server 2, and stream 4 for the travel times.

2.10. Do Prob. 1.22, except use simlib. Use stream 1 for the machine-up times and stream 2 for the repair times.

2.11. Do Prob. 1.24, except use simlib. Use stream 1 for interarrival times and stream 2 for service times. Note how much easier this model is to simulate with the list-processing tools.

2.12. Do Prob. 1.26, except use simlib. Use stream 1 for interarrival times, stream 2 for determining the customer type, stream 3 for service times of type 1 customers, and stream 4 for service times of type 2 customers.

2.13. Do Prob. 1.27, except use simlib. Use streams 1 and 2 for interarrival times and service times, respectively, for regular customers, and streams 3 and 4 for interarrival times and service times, respectively, of express customers.

2.14. Do Prob. 1.28, except use simlib. Use stream 1 for interarrival times for regular cars and stream 2 for service times for all cars.

2.15. Do Prob. 1.30, except use simlib. Use stream 1 for interarrival times, stream 2 for inspection times, stream 3 to decide whether a bus needs repair, and stream 4 for repair times.

2.16. For the inventory example of Sec. 1.5, suppose that the delivery lag is distributed uniformly between 1 and 3 months, so there could be between 0 and 3 outstanding orders at a time. Thus, the company bases its ordering decision at the beginning of each month on the sum of the (net) inventory level [denoted by $I(t)$ in Sec. 1.5] and the inventory on order; this sum could be positive, zero, or negative. For each of the nine inventory policies, run the model for 120 months and estimate the expected average total cost per month and the expected proportion of time there is a backlog. Note that holding and shortage costs are still based on the net inventory level. Use stream 1 for interdemand times, stream 2 for demand sizes, and stream 3 for delivery lags.

2.17. Problem 1.18 described a modification of the inventory system of Sec. 1.5 in which the items were perishable. Do this problem, using simlib, and in addition consider the case of LIFO (as well as FIFO) processing of the items in inventory. Use the same stream assignments as in Prob. 2.16, and in addition use stream 4 for the shelf lives.

2.18. For the time-shared computer model of Sec. 2.5, suppose that instead of processing jobs in the queue in a round-robin manner, the CPU chooses the job from the queue that has made the fewest number of previous passes through the CPU. In case of ties, the rule is FIFO. (This is equivalent to using the time of arrival to the queue to break ties.) Run the model with $n = 60$ terminals for 1000 job completions.

2.19. Ships arrive at a harbor with interarrival times that are IID exponential random variables with a mean of 1.25 days. The harbor has a dock with two berths and two cranes for unloading the ships; ships arriving when both berths are occupied join a FIFO queue. The time for one crane to unload a ship is distributed uniformly between 0.5 and 1.5 days. If only one ship is in the harbor, both cranes unload the ship and the (remaining) unloading time is cut in half. When two ships are in the harbor, one crane works on each ship. If both cranes are unloading one ship when a second ship arrives, one of the cranes immediately begins serving the second ship and the remaining service time of the first ship is doubled. Assuming that no ships are in the harbor at time 0, run the simulation for 90 days and compute the minimum, maximum, and average time that ships are in the harbor (which includes their time in berth). Also estimate the expected utilization of each berth and of the cranes. Use stream 1 for the interarrival times and stream 2 for the unloading times. [This problem is a paraphrasing of an example in Russell (1976, p. 134).]

2.20. Jobs arrive at a single-CPU computer facility with interarrival times that are IID exponential random variables with mean 1 minute. Each job specifies upon its arrival the maximum amount of processing time it requires, and the maximum times for successive jobs are IID exponential random variables with mean 1.1 minutes. However, if m is the specified maximum processing time for a particular job, the actual processing time is distributed uniformly between $0.55m$ and $1.05m$. The CPU will never process a job for more than its specified maximum; a job whose required processing time exceeds its specified maximum leaves the facility without completing service.

Simulate the computer facility until 1000 jobs have left the CPU if (*a*) jobs in the queue are processed in a FIFO manner, and (*b*) jobs in the queue are ranked in increasing order of their specified maximum processing time. For each case, compute the average and maximum delay in queue of jobs, the proportion of jobs that are delayed in queue more than 5 minutes, and the maximum number of jobs ever in queue. Use stream 1 for the interarrival times, stream 2 for the maximum processing times, and stream 3 for the actual processing times. Which operating policy would you recommend?

2.21. In a quarry, trucks deliver ore from three shovels to a single crusher. Trucks are assigned to specific shovels, so that a truck will always return to its assigned shovel after dumping a load at the crusher. Two different truck sizes are in use, 20 and 50 tons. The size of the truck affects its loading time at the shovel, travel time to the crusher, dumping time at the crusher, and return-trip time from the crusher back to its shovel, as follows (all times are in minutes):

	20-ton truck	50-ton truck
Load	Exponentially distributed with mean 5	Exponentially distributed with mean 10
Travel	Constant 2.5	Constant 3
Dump	Exponentially distributed with mean 2	Exponentially distributed with mean 4
Return	Constant 1.5	Constant 2

To each shovel are assigned two 20-ton trucks and one 50-ton truck. The shovel queues are all FIFO, and the crusher queue is ranked in decreasing order of truck size, the rule's being FIFO in case of ties. Assume that at time 0 all trucks are at their respective shovels, with the 50-ton trucks just beginning to be loaded. Run the simulation model for 8 hours and estimate the expected time-average number in queue for each shovel and for the crusher. Also estimate the expected utilizations of all four pieces of equipment. Use streams 1 and 2 for the loading times of the 20-ton and 50-ton trucks, respectively, and streams 3 and 4 for the dumping times of the 20-ton and 50-ton trucks, respectively. [This problem is taken from Pritsker (1995, pp. 153–158).]

2.22. A batch-job computer facility with a single CPU opens its doors at 7 A.M. and closes its doors at midnight, but operates until all jobs present at midnight have been processed. Assume that jobs arrive at the facility with interarrival times that are exponentially distributed with mean 1.91 minutes. Jobs request either express (class 4), normal (class 3), deferred (class 2), or convenience (class 1) service; and the classes occur with respective probabilities 0.05, 0.50, 0.30, and 0.15. When the CPU is idle, it will process the highest-class (priority) job present, the rule's being FIFO within a class. The times required for the CPU to process class 4, 3, 2, and 1 jobs are 3-Erlang random variables (see Sec. 2.7) with respective means 0.25, 1.00, 1.50, and 3.00 minutes. Simulate the computer facility for each of the following cases:

(*a*) A job being processed by the CPU is not preempted by an arriving job of a higher class.

(*b*) If a job of class i is being processed and a job of class j (where $j > i$) arrives, the arriving job preempts the job being processed. The preempted job joins the queue and takes the highest priority in its class, and only its remaining service time needs to be completed at some future time.

Estimate for each class the expected time-average number of jobs in queue and the expected average delay in queue. Also estimate the expected proportion of time that the CPU is busy and the expected proportion of CPU busy time spent on each class. Note that it is convenient to have one list for each class's queue and also an input parameter that is set to 0 for case (*a*) and 1 for case (*b*). Use stream 1 for the interarrival times, stream 2 for the job-class determination, and streams 3, 4, 5, and 6 for the processing times for classes 4, 3, 2, and 1, respectively.

2.23. A port in Africa loads tankers with crude oil for overwater shipment, and the port has facilities for loading as many as three tankers simultaneously. The tankers, which arrive at the port every 11 ± 7 hours, are of three different types. (All times given as a "±" range in this problem are distributed uniformly over the range.) The relative frequency of the various types and their loading-time requirements are:

Type	Relative frequency	Loading time, hours
1	0.25	18 ± 2
2	0.25	24 ± 4
3	0.50	36 ± 4

There is one tug at the port. Tankers of all types require the services of a tug to move from the harbor into a berth and later to move out of a berth into the harbor. When the tug is available, any berthing or deberthing activity takes about an hour. It takes the tug 0.25 hour to travel from the harbor to the berths, or vice versa, when not pulling a tanker. When the tug finishes a berthing activity, it will deberth the first tanker in the deberthing queue if this queue is not empty. If the deberthing queue is empty but the harbor queue is not, the tug will travel to the harbor and begin berthing the first tanker in the harbor queue. (If both queues are empty, the tug will remain idle at the berths.) When the tug finishes a deberthing activity, it will berth the first tanker in the harbor queue if this queue is not empty and a berth is available. Otherwise, the tug will travel to the berths, and if the deberthing queue is not empty, will begin deberthing the first tanker in the queue. If the deberthing queue is empty, the tug will remain idle at the berths.

The situation is further complicated by the fact that the area experiences frequent storms that last 4 ± 2 hours. The time between the end of one storm and the onset of the next is an exponential random variable with mean 48 hours. The tug will not start a new activity when a storm is in progress but will always finish an activity already in progress. (The berths will operate during a storm.) If the tug is traveling from the berths to the harbor without a tanker when a storm begins, it will turn around and head for the berths.

Run the simulation model for a 1-year period (8760 hours) and estimate:

(*a*) The expected proportion of time the tug is idle, is traveling without a tanker, and is engaged in either a berthing or deberthing activity

(*b*) The expected proportion of time each berth is unoccupied, is occupied but not loading, and is loading

(*c*) The expected time-average number of tankers in the deberthing queue and in the harbor queue

(*d*) The expected average in-port residence time of each type of tanker

Use stream 1 for interarrivals, stream 2 to determine the type of a tanker, stream 3 for loading times, stream 4 for the duration of a storm, and stream 5 for the time between the end of one storm and the start of the next.

A shipper considering bidding on a contract to transport oil from the port to the United Kingdom has determined that five tankers of a particular type would have to be committed to this task to meet contract specifications. These tankers would require 21 ± 3 hours to load oil at the port. After loading and deberthing, they would travel to the United Kingdom, offload the oil, return to the port for reloading, etc. The round-trip travel time, including offloading, is estimated to be 240 ± 24 hours. Rerun the simulation and estimate, in addition, the expected average in-port residence time of the proposed additional tankers. Assume that at time 0 the five additional tankers are in the harbor queue. Use the same stream assignments as before, and in addition use stream 6 for the oil-loading times at the port and stream 7 for the round-trip travel times for these new tankers. [This problem is an embellishment of one in Schriber (1974, p. 329).]

2.24. In Prob. 2.23, suppose that the tug has a two-way radio giving it the position and status of each tanker in the port. As a result, the tug changes its operating policies, as follows. If the tug is traveling from the harbor to the berths without a tanker and is less than halfway there when a new tanker arrives, it will turn around and go pick up the new tanker. If the tug is traveling from the berths to the harbor without a tanker and is less than halfway there when a tanker completes its loading, it will turn around and go pick up the loaded tanker. Run the simulation with the same parameters and stream assignments as before, under this new operating policy.

2.25. In Prob. 2.24, suppose in addition that if the tug is traveling from the harbor to the berths without a tanker and the deberthing queue is empty when a new tanker arrives, it will turn around and go pick up the new tanker, regardless of its position. Run the simulation with the same parameters and stream assignments as before, under this operating policy.

2.26. Two-piece suits are processed by a dry cleaner as follows. Suits arrive with exponential interarrival times having mean 10 minutes, and are all initially served by server 1, perhaps after a wait in a FIFO queue; see Fig. 2.67. Upon completion of service at server 1, one piece of the suit (the jacket) goes to server 2, and the other part (the pants) to server 3. During service at server 2, the jacket has a probability of 0.05 of being damaged, and while at server 3 the probability of a pair of pants being damaged is 0.10. Upon leaving server 2, the jackets go into a queue for server 4; upon leaving server 3, the pants go into a different queue for server 4. Server 4 matches and reassembles suit parts, initiating this when he is idle and two parts from the same suit are available. If both parts of the reassembled suit are undamaged, the suit is returned to the customer. If either (or both) of the parts is (are) damaged, the suit goes to customer relations (server 5). Assume that all service times are exponential, with the following means (in minutes) and use the indicated stream assignments:

Server number	Mean service time, in minutes	Stream
1	6	1
2	4	2
3	5	3
4	5 (undamaged)	4
4	8 (damaged)	5
5	12	6

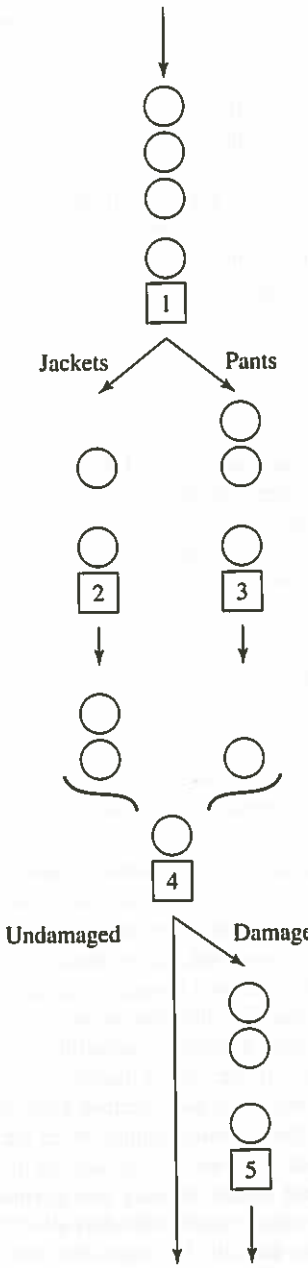

FIGURE 2.67
A dry-cleaning operation.

In addition, use stream 7 for interarrival times, and streams 8 and 9 for determining whether the pieces are damaged at servers 2 and 3, respectively. The system is initially empty and idle, and runs for exactly 12 hours. Observe the average and maximum time in the system for each type of outcome (damaged or not), separately, the average and maximum length of each queue, and the utilization of each server. What would happen if the arrival rate were to double (i.e., the interarrival-time mean were 5 minutes

instead of 10 minutes)? In this case, if you could place another person anywhere in the system to help out with one of the 5 tasks, where should it be?

2.27. A queueing system has two servers (A and B) in series, and two types of customers (1 and 2). Customers arriving to the system have their types determined immediately upon their arrival. An arriving customer is classified as type 1 with probability 0.6. However, an arriving customer may balk, i.e., may not actually join the system, if the queue for server A is too long. Specifically, assume that if an arriving customer finds m ($m \geq 0$) other customers already in the queue for A, he will join the system with probability $1/(m + 1)$, regardless of the type (1 or 2) of customer he may be. Thus, for example, an arrival finding nobody else in the queue for A (i.e., $m = 0$) will join the system for sure [probability $= 1/(0 + 1) = 1$], whereas an arrival finding 5 others in the queue for A will join the system with probability $\frac{1}{6}$. All customers are served by A. (If A is busy when a customer arrives, the customer joins a FIFO queue.) Upon completing service at A, type 1 customers leave the system, while type 2 customers are served by B. (If B is busy, type 2 customers wait in a FIFO queue.) Compute the average *total* time each type of customer spends in the *system*, as well as the number of balks. Also compute the time-average and maximum length of each queue, and both server utilizations. Assume that all interarrival and service times are exponentially distributed, with the following parameters:

- Mean interarrival time (for any customer type) = 1 minute
- Mean service time at server A (regardless of customer type) = 0.8 minute
- Mean service time at server B = 1.2 minutes

Initially the system is empty and idle, and is to run until 1000 customers (of either type) have left the system. Use stream 1 for determining the customer type, stream 2 for deciding whether a customer balks, stream 3 for interarrivals, stream 4 for service times at A (of both customer types), and stream 5 for service times at B.

2.28. An antiquated computer operates in a batch multiprocessing mode, meaning that it starts many (up to a fixed maximum of $k = 4$) jobs at a time, runs them simultaneously, but cannot start any new jobs until all the jobs in a batch are done. Within a batch, each job has its own completion time, and leaves the CPU when it finishes. There are three priority classes, with jobs of class 1 being the highest priority and class 3 jobs being the lowest priority. When the CPU finishes the last job in a batch, it first looks for jobs in the class 1 queue and takes as many as possible from it, up to a maximum of k. If there were fewer than k jobs in the class 1 queue, as many jobs as possible from the class 2 queue are taken to bring the total of class 1 and class 2 jobs to no more than the maximum batch size, k. If still more room is left in the batch, the CPU moves on to the class 3 queue. If the total number of jobs waiting in all the queues is less than k, the CPU takes them all and begins running this partially full batch; it cannot begin any jobs that subsequently arrive until it finishes all of its current batch. If no jobs at all are waiting in the queues, the CPU becomes idle, and the next arriving job will start the CPU running with a batch of size 1. Note that when a batch begins running, there may be jobs of many different classes running together in the same batch.

Within a class queue, the order of jobs taken is to be *either* FIFO or shortest job first (SJF); the simulation is to be written to handle either queue discipline by changing only an input parameter. (Thus, a job's service requirement should be generated when it arrives, and stored alongside its time of arrival in the queue. For FIFO, this would not really be necessary, but it simplifies the general programming.) The service requirement of a class i job is distributed uniformly between constants $a(i)$ and $b(i)$ minutes. Each class has its own separate arrival process, i.e., the interarrival time between two successive class i jobs is exponentially distributed with mean $r(i)$ minutes. Thus, at any given point in the simulation, there should be three separate arrivals scheduled, one for each class. If a job arrives to find the CPU busy, it joins the queue for its class in the appropriate place, depending on whether the FIFO or SJF option is in force. A job arriving to find the CPU idle begins service immediately; this would be a batch of size 1. The parameters are as follows:

i	$r(i)$	$a(i)$	$b(i)$
1	0.2	0.05	0.11
2	1.6	0.94	1.83
3	5.4	4.00	8.00

Initially the system is empty and idle, and the simulation is to run for exactly 720 minutes. For each queue, compute the average, minimum, and maximum delay, as well as the time-average and maximum length. Also, compute the utilization of the CPU, defined here as the proportion of time it is busy regardless of the number of jobs running. Finally, compute the time-average number of jobs running in the CPU (where 0 jobs are considered running when the CPU is idle). Use streams 1, 2, and 3 for the interarrival times of jobs of class 1, 2, and 3, respectively, and streams 4, 5, and 6 for their respective service requirements. Suppose that a hardware upgrade could increase k to 6. Would this be worth it?

2.29. Consider a queueing system with a fixed number $n = 5$ of parallel servers fed by a single queue. Customers arrive with interarrival times that are exponentially distributed with mean 5 (all times are in minutes). An arriving customer finding an idle server will go directly into service, choosing the leftmost idle server if there are several, while an arrival finding all servers busy joins the end of the queue. When a customer (initially) enters service, her service requirement is distributed uniformly between $a = 2$ and $b = 2.8$, but upon completion of her initial service, she may be "dissatisfied" with her service, which occurs with probability $p = 0.2$. If the service was satisfactory, the customer simply leaves the system, but if her service was not satisfactory, she will require further service. The determination as to whether a service was satisfactory is to be made when the service is completed. If an unsatisfactory service is completed and there are no other customers waiting in the queue, the dissatisfied customer immediately begins another service time at her same server. On the other hand, if there is a queue when an unsatisfactory service is completed, the dissatisfied customer must join the queue (according to one of two options, described below), and the server takes the first person from the queue to serve next. Each time a customer *re*enters service, her service time and probability of being dissatisfied are lower; specifically, a customer who has *already* had i (unsatisfactory) services has a next service time that is distributed uniformly between $a/(i + 1)$ and $b/(i + 1)$, and her probability of being dissatisfied with this next service is $p/(i + 1)$. Theoretically, there is no upper limit on the number of times a given customer will have to be served to be finally satisfied.

There are two possible rules concerning what to do with a dissatisfied customer when other people are waiting in queue; the program is to be written so that respecifying

a single input parameter will change the rule from (i) to (ii):

(i) A customer who has just finished an unsatisfactory service joins the end of the queue.

(ii) A customer who has just finished an unsatisfactory service rejoins the queue so that the next person taken from the (front of the) queue will be the customer who has already had the largest number of services; the rule is FIFO in case of ties. This rule is in the interest of both equity and efficiency, since customers with a long history of unsatisfactory service tend to require shorter service and also tend to be more likely to be satisfied with their next service.

Initially the system is empty and idle, and the simulation is to run for exactly 480 minutes. Compute the average and maximum total time in system [including all the delay(s) in queue and service time(s) of a customer], and the number of satisfied customers who leave the system during the simulation. Also compute the average and maximum length of the queue, and the time-average and maximum number of servers that were busy. Use stream 1 for interarrivals, stream 2 for all service times, and stream 3 to determine whether each service was satisfactory.

2.30. The student-center cafeteria at Big State University is trying to improve its service during the lunch rush from 11:30 A.M. to 1:00 P.M. Customers arrive together in groups of size 1, 2, 3, and 4, with respective probabilities 0.5, 0.3, 0.1, and 0.1. Interarrival times between groups are exponentially distributed with mean 30 seconds. Initially, the system is empty and idle, and is to run for the 90-minute period. Each arriving customer, whether alone or part of a group, takes one of three routes through the cafeteria (groups in general split up after they arrive):

- Hot-food service, then drinks, then cashier
- Specialty-sandwich bar, then drinks, then cashier
- Drinks (only), then cashier

The probabilities of these routes are respectively 0.80, 0.15, and 0.05; see Fig 2.68. At the hot-food counter and the specialty-sandwich bar, customers are served one at a time (although there might actually be one or two workers present, as discussed below). The drinks stand is self-service, and assume that nobody ever has to queue up here; this is equivalent to thinking of the drinks stand as having infinitely many servers. There are either two or three cashiers (see below), each having his own queue, and there is no jockeying; customers arriving to the cashiers simply choose the shortest queue. All queues in the model are FIFO.

In Fig. 2.68, ST stands for service time at a station, and ACT stands for the accumulated (future) cashier time due to having visited a station; the notation ~U(a, b) means that the corresponding quantity is distributed uniformly between a and b seconds. For example, a route 1 customer goes first to the hot-food station, joins the queue there if necessary, receives service there that is uniformly distributed between 50 and 120 seconds, "stores away" part of a (future) cashier time that is uniformly distributed between 20 and 40 seconds, then spends an amount of time uniformly distributed between 5 seconds and 20 seconds getting a drink, and accumulates an additional amount of (future) cashier time distributed uniformly between 5 seconds and 10 seconds. Thus, his service requirement at a cashier will be the sum of the U(20, 40) and U(5, 10) random variates he "picked up" at the hot-food and drinks stations.

Report the following measures of system performance:

- The average and maximum delays in queue for hot food, specialty sandwiches, and cashiers (regardless of which cashier)

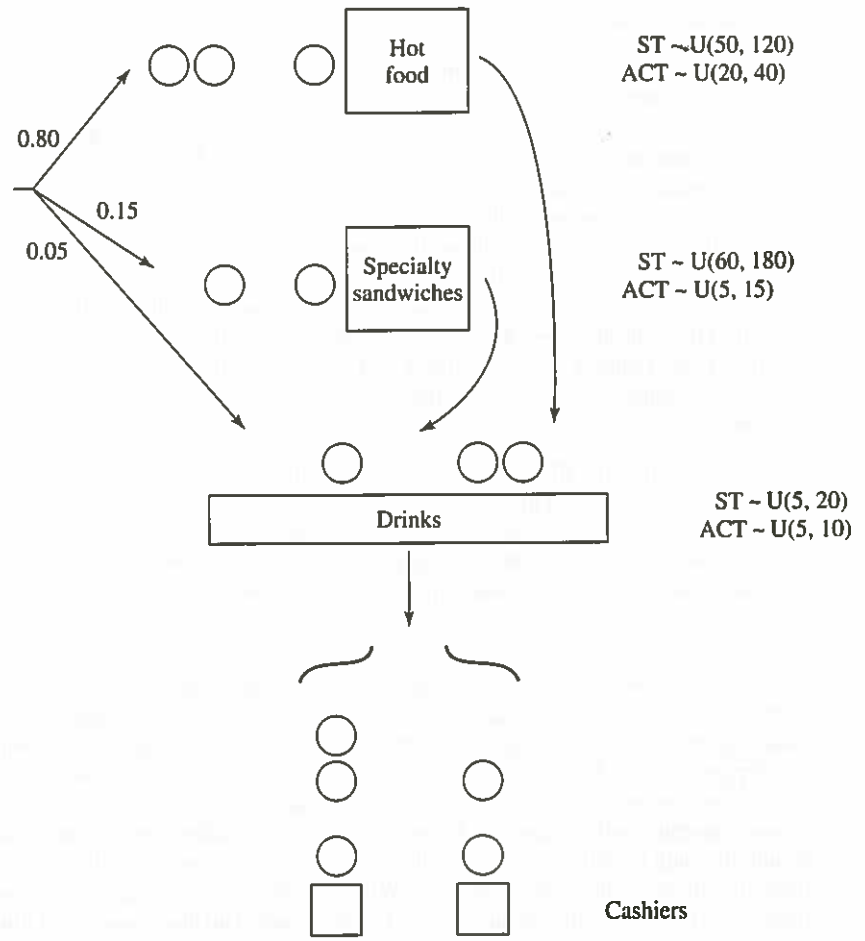

FIGURE 2.68
The BSU cafeteria.

- The time-average and maximum number in queue for hot food and specialty sandwiches (separately), and the time-average and maximum total number in all cashier queues
- The average and maximum total delay in all the queues for each of the three types of customers (separately)
- The overall average total delay for all customers, found by weighting their individual average total delays by their respective probabilities of occurrence
- The time-average and maximum total number of customers in the entire system (for reporting to the fire marshall)

There are several questions about the system's operation. For security reasons, there must be at least 2 cashiers, and the maximum number of cashiers is 3. Also, there must be at least one person working at each of the hot-food and specialty-sandwich stations. Thus, the minimum number of employees is 4; run this as the "base-case" model.

Then, consider adding employees, in several ways:

(*a*) Five employees, with the additional person used in one of the following ways:
  (i) As a third cashier
  (ii) To help at the hot-food station. In this case, customers are still served one at a time, but their service time is cut in half, being distributed uniformly between 25 seconds and 60 seconds.
  (iii) To help at the specialty-sandwich bar, meaning that service is still one at a time, but distributed uniformly between 30 seconds and 90 seconds
(*b*) Six employees, in one of the following configurations:
  (i) Two cashiers, and two each at the hot-food and specialty-sandwich stations
  (ii) Three cashiers, two at hot food, and one at specialty sandwiches
  (iii) Three cashiers, one at hot food, and two at specialty sandwiches
(*c*) Seven employees, with three cashiers, and two each at the hot-food and specialty-sandwich stations

Run the simulation for all seven expansion possibilities, and make a recommendation as to the best employee deployment at each level of the number of employees. In all cases, use stream 1 for the interarrival times between groups, stream 2 for the group sizes, stream 3 for an individual's route choice, streams 4, 5, and 6 for the STs at the hot-food, specialty-sandwich, and drinks stations, respectively, and streams 7, 8, and 9 for the ACTs at these respective stations.

2.31. Consolidated Corkscrews (CC) is a multinational manufacturer of precision carbon-steel corkscrews for heavy-duty, high-speed use. Each corkscrew is made on a metal lathe, and in order to meet rising consumer demand for their product, CC is planning a new plant with six lathes. They are not sure, however, how this new plant should be constructed, or how the maintenance department should be equipped. Each lathe has its own operator, who is also in charge of repairing the lathe when it breaks down. Reliability data on lathe operation indicate that the "up" time of a lathe is exponentially distributed with mean 75 minutes. When a lathe goes down, its operator immediately calls the tool crib to request a tool kit for repairs. The plant has a fixed number, m, of tool kits, so there may or may not be a kit in the crib when an operator calls for one. If a tool kit is not available, the operator requesting one is placed in a FIFO queue and must wait his or her turn for a kit; when one later becomes available, it is then placed on a conveyor belt and arrives t_i minutes later to lathe i, where t_i might depend on the lathe number, i, requesting the kit. If a kit is available, it is immediately placed on a conveyor belt and arrives at the broken lathe t_i minutes later; in this case the operator's queue delay is counted as 0. When an operator of a broken lathe receives a tool kit, he or she begins repair, which takes an amount of time distributed as a 3-Erlang random variable with mean 15 minutes. When the repair is complete, the lathe is brought back up and the tool kit is sent back to the tool crib, where it arrives t_i minutes later, if it is sent back from lathe i. Initially, assume that all lathes are up and have just been "freshly repaired," and that all m tool kits are in the crib. CC wants to know about the projected operation of the plant over a continuous 24-hour day by looking at:

- The proportion of time that each of the six lathes is down
- The time-average number of lathes that are down
- The time-average number of tool kits sitting idle in the crib
- The average delay in queue of operators requesting a tool kit

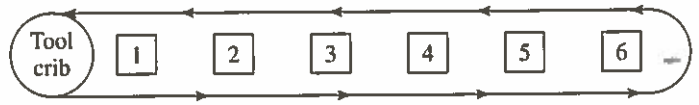

FIGURE 2.69
The linear design.

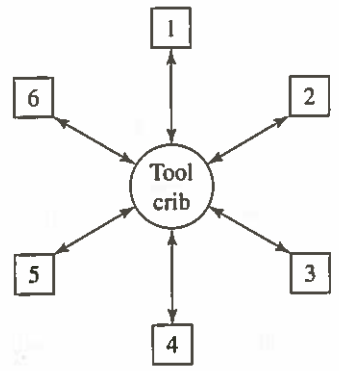

FIGURE 2.70
The circular design.

There are two major questions to be addressed:

(*a*) How should the plant be laid out? Two layouts are under consideration:
  (i) In the linear design (see Fig. 2.69), the lathes are placed in a straight line with the tool crib at the left end, and a single conveyor belt for the tool kits can reach all lathes. In this case, $t_i = 2i$ minutes, for $i = 1, 2, \ldots, 6$.
  (ii) In the circular design, the lathes are placed around the tool crib (see Fig. 2.70), and each lathe has its own conveyor belt to the crib; here, $t_i = 3$ for all lathe numbers i. This is a more expensive design, but results in shorter travel times for the kits.
(*b*) How many tool kits should there be? As tool kits are quite expensive, CC does not want to purchase more than necessary.

Carry out the necessary simulations and advise CC on questions (*a*) and (*b*). In all cases, use stream 1 for the lathe-up times, and stream 2 for repair times.

2.32. The engines on jet aircraft must be periodically inspected and, if necessary, repaired. An inspection/repair facility at a large airport handles seven different types of jets, as described in the table below. The times between successive arrivals of planes of type i (where $i = 1, 2, \ldots, 7$) are exponentially distributed with mean $a(i)$, as given in the table; all times are in days. There are n parallel service stations, each of which sequentially handles the inspection and repair of all the engines on a plane, but can deal with only one engine at a time. For example, a type 2 plane has three engines, so when it enters service, each engine must undergo a complete inspection and repair process (as described below) before the next engine on this plane can begin service, and all three engines must be inspected and (if necessary) repaired before the plane leaves the service station. Each service station is capable of dealing with any type of plane. As usual, a plane arriving to find an idle service station goes directly into service, while an arriving plane finding all service stations occupied must join a single queue.

Plane type (i)	Number of engines	$a(i)$	$A(i)$	$B(i)$	$p(i)$	$r(i)$	$c(i)$
1	4	8.1	0.7	2.1	0.30	2.1	2.1
2	3	2.9	0.9	1.8	0.26	1.8	1.7
3	2	3.6	0.8	1.6	0.18	1.6	1.0
4*	4	8.4	1.9	2.8	0.12	3.1	3.9
5	4	10.9	0.7	2.2	0.36	2.2	1.4
6	2	6.7	0.9	1.7	0.14	1.7	1.1
7*	3	3.0	1.6	2.0	0.21	2.8	3.7

Two of the seven types of planes are classified as widebody (denoted by an asterisk * in the above table), while the other five are classified as regular. Two disciplines for the queue are of interest:

(i) Simple FIFO with all plane types mixed together in the same queue

(ii) Nonpreemptive priority given to widebody jets, with the rule being FIFO within the widebody and regular classifications

For each engine on a plane (independently), the following process takes place (i denotes the plane type):

- The engine is initially inspected, taking an amount of time distributed uniformly between $A(i)$ and $B(i)$.
- A decision is made as to whether repair is needed; the probability that repair is needed is $p(i)$. If no repair is needed, inspection of the jet's next engine begins; or if this was the last engine, the jet leaves the facility.
- If repair is needed, it is carried out, taking an amount of time distributed as a 2-Erlang random variable with mean $r(i)$.
- After repair, another inspection is done, taking an amount of time distributed uniformly between $A(i)/2$ and $B(i)/2$ (i.e., half as long as the initial inspection, since tear-down is already done). The probability that the engine needs further repair is $p(i)/2$.
- If the initial repair was successful, the engine is done. If the engine still fails inspection, it requires further repair, taking an amount of time distributed as 2-Erlang with mean $r(i)/2$, after which it is inspected again, taking an amount of time distributed uniformly between $A(i)/2$ and $B(i)/2$; it fails this inspection with probability $p(i)/2$, and would need yet more repair, which would take a 2-Erlang amount of time with mean $r(i)/2$. This procedure continues until the engine finally passes inspection. The mean repair time stays at $r(i)/2$, the probability of failure stays at $p(i)/2$, and the inspection times stay between $A(i)/2$ and $B(i)/2$.

A cost of $c(i)$ (measured in tens of thousands of dollars) is incurred for every (full) day a type i plane is down, i.e., is in queue or in service. The general idea is to study how the total (summed across all plane types) average daily downtime cost depends on the number of service stations, n. Initially the system is empty and idle, and the simulation is to run for 365 round-the-clock days. Observe the average delay in queue for each plane type and the overall average delay in queue for all plane types, the time-average number of planes in queue, the time-average number of planes down for each plane type separately, and the total average daily downtime cost for all planes added together. Try various values of n to get a feel for the system's behavior. Recommend a choice for n, as well as which of the queue disciplines (i) or (ii) above appears to lead to the most cost-effective operation. Use streams 1 through 7 for the interarrival times of plane types $i = 1$ through $i = 7$, respectively, streams 8 through 14 for their respective inspection times (first or subsequent), streams 15 through 21 to determine whether they need (additional) repair, and streams 22 through 28 for their repair times (first or subsequent).

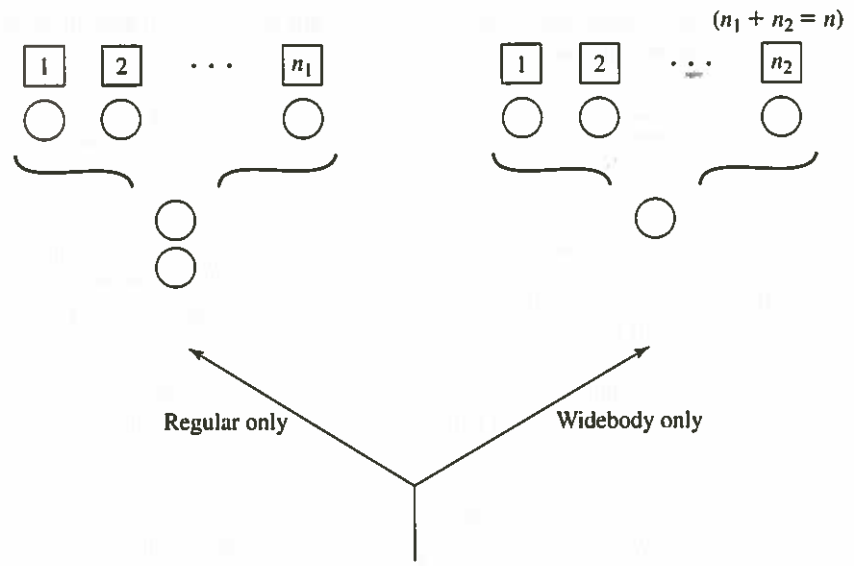

FIGURE 2.71
Alternative layout for the aircraft-repair facility.

As an alternative to the above layout, consider separating entirely the service of the widebody and regular jets. That is, take n_2 of the n stations and send all the widebody jets there (with a single queue of widebodies feeding all n_2 stations), and the remaining $n_1 = n - n_2$ stations are for regular jets only; see Fig. 2.71. Do you think that this alternative layout would be better? Why? Use the same parameters and stream assignments as above.

2.33. Write a C function "delete" to delete the (logically) first record from list "list" with a value "value" (a float-valued representation of an integer, for example, 5.0 to represent 5) for attribute "attribute". Place the attributes of the deleted record in the transfer array. To delete a desired record, a statement of the form "delete(list, value, attribute)" should be executed. If there is an error condition (e.g., there is no matching record in the list), return a value of 0; otherwise return a value of 1. Also, update the statistics for list "list" by invoking timest (see the code for function remove in App. 2A).

2.34. Write a C function "insert" to insert a new event record into the event list, using the middle-pointer algorithm discussed in Sec. 2.8. If two event records have the same event time, give preference to the event with the lowest-numbered event type.

2.35. For the bank model in Sec. 2.6, suppose that after a customer has waited in queue a certain amount of time, the customer *may* leave without being served; this is called *reneging*. Assume that the amount of time a customer will wait in queue before considering reneging is distributed uniformly between 5 and 10 minutes; if this amount of

time does elapse while the customer is in queue, the customer will actually leave with the following probabilities:

Position in queue when time elapses	1	2	3	≥ 4
Probability of reneging	0.00	0.25	0.50	1.00

Using the function "delete" from Prob. 2.33, run the simulation model with five tellers and estimate (in addition to what was estimated before) the expected proportion of customers who renege and the expected average delay in queue of the reneging customers. Use the same stream assignments as in Sec. 2.6, and in addition use stream 3 for the time a customer will wait in queue before considering reneging, and stream 4 for determining if he or she actually reneges if this time elapses.

2.36 A five-story office building is served by a single elevator. People arrive to the ground floor (floor 1) with independent exponential interarrival times having mean 1 minute. A person will go to each of the upper floors with probability 0.25. It takes the elevator 15 seconds to travel one floor. Assume, however, that there is no loading or unloading time for the elevator at a particular floor. A person will stay on a particular floor for an amount of time that is distributed uniformly between 15 and 120 minutes. When a person leaves floor i (where $i = 2, 3, 4, 5$), he or she will go to floor 1 with probability 0.7, and will go to each of the other three floors with probability 0.1. The elevator can carry six people, and starts on floor 1. If there is not room to get all people waiting at a particular floor on the arriving elevator, the excess remain in queue. A person coming down to floor 1 departs from the building immediately. The following control logic also applies to the elevator:

- When the elevator is going up, it will continue in that direction if a current passenger wants to go to a higher floor or if a person on a higher floor wants to get on the elevator.
- When the elevator is going down, it will continue in that direction if it has at least one passenger or if there is a waiting passenger at a lower floor.
- If the elevator is at floor i (where $i = 2, 3, 4$) and going up (down), then it will not immediately pick up a person who wants to go down (up) at that floor.
- When the elevator is idle, its home base is floor 1.
- The elevator decides at each floor what floor it will go to next. It will not change directions between floors.

Use the following random-number stream assignments:

1, interarrival times of people to the building
2, next-floor determination (generate upon arrival at origin floor)
3, length of stay on a particular floor (generate upon arrival at floor)

Run a simulation for 20 hours and gather statistics on:

(*a*) Average delay in queue in each direction (if appropriate), for each floor
(*b*) Average of individual delays in queue over all floors and all people
(*c*) Proportion of time that the elevator is moving with people, is moving empty, and is idle (on floor 1)
(*d*) Average and maximum number in the elevator
(*e*) Proportion of people who cannot get on the elevator since it is full, for each floor

Rerun the simulation if the home base for the elevator is floor 3. Which home base gives the smallest average delay [output statistic (*b*)]?

2.37. Coal trains arrive to an unloading facility with independent exponential interarrival times with mean 10 hours. If a train arrives and finds the system idle, the train is unloaded immediately. Unloading times for the train are independent and distributed uniformly between 3.5 and 4.5 hours. If a train arrives to a busy system, it joins a FIFO queue.

The situation is complicated by what the railroad calls "hogging out." In particular, a train crew can work for only 12 hours, and a train cannot be unloaded without a crew present. When a train arrives, the remaining crew time (out of 12 hours) is independent and distributed uniformly between 6 and 11 hours. When a crew's 12 hours expire, it leaves immediately and a replacement crew is called. The amount of time between when a replacement crew is called and when it actually arrives is independent and distributed uniformly between 2.5 and 3.5 hours.

If a train is being unloaded when its crew hogs out, unloading is suspended until a replacement crew arrives. If a train is in queue when its crew hogs out, the train cannot leave the queue until its replacement crew arrives. Thus, the unloading equipment can be idle with one or more trains in queue.

Run the simulation for 720 hours (30 days) and gather statistics on:
(*a*) Average and maximum time a train spends in the system
(*b*) Proportion of time unloading equipment is busy, idle, and hogged out
(*c*) Average and maximum number of trains in queue
(*d*) Proportion of trains that hog out 0, 1, and 2 times

Note that if a train is in queue when its crew hogs out, the record for this train must be accessed. (This train may be anywhere in the queue.) Use the C function "delete" from Prob. 2.33.

2.38. Consider a car-rental system shown in Fig. 2.72, with all distances given in miles. People arrive at location i (where $i = 1, 2, 3$) with independent exponential interarrival times at respective rates of 14, 10, and 24 per hour. Each location has a FIFO queue with unlimited capacity. There is one bus with a capacity of 20 people and a speed of 30 miles per hour. The bus is initially at location 3 (car rental), and leaves immediately in a counterclockwise direction. All people arriving at a terminal want to go to the car rental. All people arriving at the car rental want to go to terminals 1 and 2 with

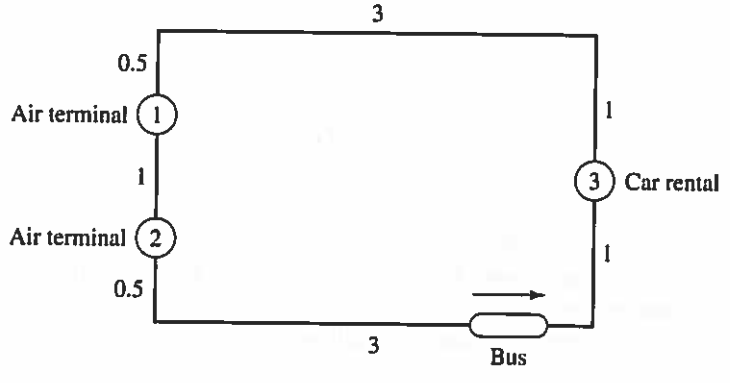

FIGURE 2.72
A car rental system.

respective probabilities 0.583 and 0.417. When a bus arrives at a location, the following rules apply:

- People are first unloaded in a FIFO manner. The time to unload one person is distributed uniformly between 16 and 24 seconds.
- People are then loaded on the bus up to its capacity, with a loading time per person that is distributed uniformly between 15 and 25 seconds.
- The bus always spends at least 5 minutes at each location. If no loading or unloading is in process after 5 minutes, the bus will leave immediately.

Run a simulation for 80 hours and gather statistics on:
(*a*) Average and maximum number in each queue
(*b*) Average and maximum delay in each queue
(*c*) Average and maximum number on the bus
(*d*) Average, maximum, and minimum time the bus is stopped at each location
(*e*) Average, maximum, and minimum time for the bus to make a loop (departure from the car rental to the next such departure)
(*f*) Average, maximum, and minimum time a person is in the system by arrival location

Use the following random-number stream assignments:
i, interarrival times at location *i* (where $i = 1, 2, 3$)
4, unloading times
5, loading times
6, determining destination of an arrival at the car rental

CHAPTER 3

Simulation Software

Recommended sections for a first reading: 3.1 through 3.4

3.1 INTRODUCTION

In studying the simulation examples in Chaps. 1 and 2, the reader probably noticed several features needed in programming most discrete-event simulation models, including:

- Generating random numbers, that is, observations from a U(0,1) probability distribution
- Generating random variates from a specified probability distribution (e.g., exponential)
- Advancing simulated time
- Determining the next event from the event list and passing control to the appropriate block of code
- Adding records to, or deleting records from, a list
- Collecting output statistics and reporting the results
- Detecting error conditions

As a matter of fact, it is the commonality of these and other features to most simulation programs that led to the development of special-purpose simulation packages. Furthermore, we believe that the improvement and greater ease of use of these packages have been major factors in the increased popularity of simulation in recent years.

We discuss in Sec. 3.2 the relative merits of using a simulation package rather than a programming language such as C, C++, or Java for building simulation models. In Sec. 3.3 we present a classification of simulation software, including