1. (40 points) The first four customers enter and exit a queueing system at the following times:

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | Enter Time | Exit Time | **Time in System** |
| 1 | 18.5 | 30.1 | **11.6** |
| 2 | 25.6 | 34.2 | **8.6** |
| 3 | 27.3 | 45.7 | **18.4** |
| 4 | 31.2 | 43.5 | **12.3** |
|  |  | **Total** | **50.9** |

* 1. Explicitly compute the average time in the system for the first four customers. Assume there are no other customers in the system during the times considered.

**Average time in system = 50.9/4 =12.75**

* 1. Explicitly (i.e., *without* using Little’s formula) compute the average number in the system at the time 45.0. (*Hint*: consider a state trajectory).

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Event | # in System | Area |
| 0.0 | Run | 0 | 0.0 |
| 18.5 | Arrival | 1 | 0.0 |
| 25.6 | Arrival | 2 | 7.1 |
| 27.3 | Arrival | 3 | 10.5 |
| 30.1 | EndService | 2 | 18.9 |
| 31.2 | Arrival | 3 | 21.1 |
| 34.2 | EndService | 2 | 30.1 |
| 43.5 | EndService | 1 | 48.7 |
| 45.0 |  | 1 | 50.2 |

**Average # in System at time 45.0 = 50.2/45.0 ≅ 1.1156**

* 1. For the system in problem 1 which of the following statements is true (circle all statements that are true)?
     1. The system must have exactly one server
     2. The system must have exactly two servers
     3. **The system must have more than one server**
     4. It is impossible to tell whether there are one or more than one servers from the data.
  2. Justify your answer in part c above.

**Customer 4 entered after customer 3 but completed service before customer 3. Therefore, there must be more than 1 server. There could be more than 2 servers, however.**

1. (25 points) Following is the state of a simulation run for the finite capacity queueing model with a capacity of 1 (i.e., c = 1).[[1]](#footnote-1) Perform the next four iterations of the Event List algorithm. Use the following values for interarrival and service times: {*tA*} = {2.8, 4.1, 5.6, …}, {*tS*} = {2.4, 5.2, …} (you will not have to use all the data). **NOTE: the first event should have been JoinQueue.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Current Time | Current Event | Q | S | N | L | Event List | |
| 5.2 | Join Queue | 1 | 0 | 5 | 0 | 5.8 8.2 10.1 | Arrival EndService EndService |
| 5.8 | Arrival | 1 | 0 | 6 | 0 | 5.8 8.2 8.6 10.1 | Leave EndService Arrival EndService |
| 5.8 | Leave | 1 | 0 | 6 | 1 | 8.2 8.6 10.1 | EndService Arrival EndService |
| 8.2 | EndService | 1 | 1 | 6 | 1 | 8.2 8.6 10.1 | StartService Arrival EndService |
| 8.2 | StartService | 0 | 0 | 6 | 1 | 8.6 10.1 10.6 | Arrival EndService EndService |

1. (35 points) Ceramic parts arrive one at a time to a single kiln[[2]](#footnote-2) according to an arrival process with interarrival times of {*tA*}. The parts are processed in batches of fixed size, *B*. Each batch requires a random amount {*tS*}of time to process; only batches of exactly size *B* are processed – no partial batches. That is, the kiln will be started only when there are *B* (or more) parts available. When a kiln is processing a batch of parts, it cannot be used for anything else. After *m* batches have been processed, maintenance must be performed on the kiln by the operator, taking *tM* time units. During maintenance the kiln cannot be used to fire parts.

Formulate an Event Graph model for this system. You do not have to explicitly show how any measures are to be estimated. Be sure to define your parameters and state variables and draw your Event Graph clearly. *Note*: you do *not* have to use listeners for this model.This page is for your work

|  |  |
| --- | --- |
| **Parameters** | **State Variables** |
| {tA} = interarrival times | Q = # in queue (0) |
| {tB} = batch processing times | S = 1 if kiln available, 0 if firing |
| B = batch size | N = # batches since last maintenance |
| m = # batches before maintenance |  |
| {tM}= maintenance times |  |

**Event Graph:**



1. (20 Points) For the finite capacity queue shown on the last page, write the Simkit code for (just) the Arrival event. Use the following variable names:

{tA} = interarrivalTimeGenerator Q = numberInQueue

{tS} = serviceTimeGenerator S = numberAvailableServers

c = capacity N = numberArrivals

k = totalNumberServers L = numberLost

Assume that any getters/setters have been defined. You may use “fpc” as an abbreviation for “firePropertyChange” and “fipc” for “fireIndexedPropertyChange” if needed.

public void doArrival() {

int oldNumberArrivals = getNumberArrivals();

numberArrivals += 1;

firePropertyChange(“numberArrivals”, oldNumberArrivals, getNumberArrivals();

waitDelay(“Arrival”, interarrivlaTimeGenerator);

if (numberInQueue < capacity) {

waitDelay(“JoinQueue”, 0.0);

}

If (numberInQueue == capacity) {

waitDelay(“Leave”, 0.0);

}

}

1. See the last page for the Event Graph of this model. Note that L is the number of customers who leave due to the queue being full. [↑](#footnote-ref-1)
2. A kiln is a furnace or oven for burning, baking, or drying. [↑](#footnote-ref-2)