1. This problem concerns the Trivedi cell phone service model, pp.202ff in our book. Of course, use the same notation that is used in the book.

(a) (20) Give the balance equation for state 0, i.e. (11.5) for the case $i = 0$.

(b) (20) As we progress through time, the cell will alternate between what we will call AllCallsBlocked and NewCallsOK periods. Find the mean length of AllCallsBlocked periods.

2. (20) Suppose the k-element vector X has mean vector μ and covariance matrix Σ. Define the scalar Y to be the difference between the first element of X and the average of the remaining elements ("difference" meaning the former minus the latter). Find $Var(Y)$, expressed *only* in terms of μ, Σ, and k. You may use ellipsis ("...") notation, but may NOT refer to individual elements of μ or Σ.

3. (20) In this problem you will write an R function **mcsim(p,nsteps)** whose purpose it is to find the approximate π vector for a discrete-time Markov chain via simulation. It would be used mainly for infinite-state space settings. Here are the details:

- The argument **p** is a user-supplied function that specifies the transition matrix; e.g. **p(2,5)** returns the probability of going from state 2 to state 5 (in one step).
- The argument **nsteps** is the number of time steps to be simulated, i.e. $X_1, X_2, ..., X_{\text{nsteps}}$.
- The return value will be the approximate π vector, expressed as an R list, with a component for each nonzero value of our approximate π. For instance, if we name our list **pi** and find that π_2 and π_4 are approximately 0.8 and 0.2, then we will set **pi**[[**2**]] and **pi**[[**4**]] to 0.8 and 0.2.
- Use the following method. First, use simulation to determine whether our new state is 1 or not. If not, then simulate to determine whether our new state is 2 or not, and so on.

If you are not very familiar with R lists, try typing this code into R's interactive mode:

```
z <- list()
z[[3]] <- 8
z
i <- 6
z[[i]] <- 8888
z
```

Solutions:

1.a

$$\pi_0 \lambda = \pi_1 \mu$$

1.b (It was explained during the quiz that AllCallsBlocked periods consist of times during which NO call not already in service will be accepted, no matter whether handoff call or new call originating within the cell.)

Picture what happens when we enter such a period. There had been one free channel, but now none is free. The period will end when some call currently in service ends, which could occur either by someone hanging up or by a caller leaving the cell. Those events are occurring for each caller at rate μ, for a total rate of $n\mu$. Moreover, the time passing before such a transition has an exponential distribution (due to the fact that a minimum of independent exponentially-distributed random variables itself has an exponential distribution).

In other words, the mean length of the period will be $1/(n\mu)$.

2. Define a k-component random vector

$$u = (1, -1/(k-1), -1/(k-1), ..., -1/(k-1))'$$

Then

$$Y = u'X$$

Thus

$$Var(Y) = Var(u'X) = u'\Sigma u$$

by (12.54).

3.

```
# find the approximate stationary distribution pi for a discrete-time
# Markov chain, using simulation; intended mainly for infinite state
# spaces

# states are 1,2,3,...

# arguments:

#    p(i,j): user-defined, returns element (i,j) of the transition
#            matrix
#    nsteps: number of time steps to simulate

# value:

#    an R list, one element for each nonzero element of pi

mcsim <- function(p,nsteps) {
   # record number of visits to each state found so far
   visits <- list()
   currstate <- 1  # arbitrary starting state
   visits[[currstate]] <- 1
   for (i in 1:nsteps) {
      # get next state
      currstate <- simnextj(currstate,p)
      # R lists are touchy about adding new elements, so updating visits
      # list is delicate
      if (currstate > length(visits))
         visits[[currstate]] <- 1 else
         if (is.null(visits[[currstate]]))
            visits[[currstate]] <- 1 else
            visits[[currstate]] <- visits[[currstate]] + 1
   }
   # sim done, change to proportions
   for (i in 1:length((visits))) {
      tmp <- visits[[i]]
      if (!is.null(tmp))
         visits[[i]] <- visits[[i]] / nsteps
   }
   visits
}

# simulate next state, given currently at i
simnextj <- function(i,p) {
   j <- 1
   tot <- 0
   repeat {
      pij <- p(i,j)
      # must use CONDITIONAL probability
      if (pij > 0 && runif(1) <= pij / (1-tot))
         return(j)
      tot <- tot + pij
      if (tot >= 1) return(j)
      j <- j + 1
   }
}

# test example: keep flipping coin; each time get 3 consecutive heads,
# win prize; state i means i-1 consecutive heads so far (should be 0,1,2
# but to have indices start at 1, have 1,2,3)

consec3 <- function(i,j) {
   if (i==1 && j == 1 ||
        i==1 && j == 2 ||
        i==2 && j == 3 ||
        i==2 && j == 1) return(0.5)
   if (i==3 && j == 1) return(1)
   return(0)
}

# test with mcsim(consec3,1000)
```