1.

Use the Chapman-Kolmogorov Equation:

\[ P_{ij}^n = \sum_k P_{ik}^{n-r} P_{kj}^r > 0 \]

2.

a.

```r
library(markovchain)
# Transition matrix
tmA1 <- matrix(c(0,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0), nrow=3, byrow=TRUE);
tmA1

[,1] [,2] [,3]
[1,] 0.0 0.5 0.5
[2,] 0.5 0.0 0.5
[3,] 0.5 0.5 0.0

# Create DTMC:
dtmcA1 <- as(tmA1, "markovchain");
summary(dtmcA1)
```

Unnamed Markov chain  Markov chain that is composed by:
Closed classes:
s1 s2 s3
Recurrent classes:
{s1,s2,s3}
Transient classes:
NONE
The Markov chain is irreducible
The absorbing states are: NONE

Recurrent classes: \{s1,s2,s3\} therefore, all the three states are closed under one recurrent class making the Markov Chain is recurrent.
b.

```r
# Transition matrix
tmA2 <- matrix(c(0,0,1,0,0,0,1,0.5,0,0,0,0,1,0,1,0), nrow=4, byrow=TRUE);
tmA2

[1,] 0.0 0.0 0 1
[2,] 0.0 0.0 0 1
[3,] 0.5 0.5 0 0
[4,] 0.0 0.0 1 0

# Create DTMC:
dtmcA2 <- as(tmA2, "markovchain");
summary(dtmcA2)
```

Unnamed Markov chain Markov chain that is composed by:
Closed classes:
s1 s2 s3 s4
Recurrent classes:
{s1,s2,s3,s4}
Transient classes:
NONE
The Markov chain is irreducible
The absorbing states are: NONE

Recurrent classes: {s1,s2,s3, s4} therefore, all the four states are closed under one recurrent class making the Markov Chain is recurrent.

c.

```r
# Transition matrix
tmA3 <- matrix(c(0.5,0,0.5,0,0,0.25,0.5,0.5,0.25,0,0,0.5,0,0.5,0,0.5,0.5), nrow=5, byrow=TRUE);
tmA3

[1,] 0.5 0.0 0.5 0.0 0.0
[2,] 0.25 0.5 0.25 0.0 0.0
[3,] 0.5 0.0 0.5 0.0 0.0
[4,] 0.0 0.0 0.0 0.5 0.5
[5,] 0.0 0.0 0.0 0.5 0.5

# Create DTMC:
dtmcA3 <- as(tmA3, "markovchain");
```
summary(dtmcA3)

Unnamed Markov chain  Markov chain that is composed by:
Closed classes:
s1  s3
s4  s5
Recurrent classes:
{s1,s3},{s4,s5}
Transient classes:
{s2}
The Markov chain is not irreducible
The absorbing states are: NONE

Recurrent classes: {s1,s3},{s4,s5} and Transient classes: {s2}

d.

# Transition matrix
tmA4 <- matrix(c(0.25,0.75,0,0,0,0.5,0.5,0,0,0,
                 0,0,1,0,0,0,0.3333333,0.6666667,0,
                 0,0,0,0.3333333,0.6666667,0,
                 0,0,0,1,0,0,0,0), nrow=5, byrow=TRUE);
tmA4

[1,] 0.25 0.75 0.0000000 0.0000000 0
[2,] 0.50 0.50 0.0000000 0.0000000 0
[3,] 0.00 0.00 1.0000000 0.0000000 0
[4,] 0.00 0.00 0.3333333 0.6666667 0
[5,] 1.00 0.00 0.0000000 0.0000000 0

# Create DTMC:
dtmcA4 <- as(tmA4, "markovchain")
summary(dtmcA4)

Unnamed Markov chain  Markov chain that is composed by:
Closed classes:
s1   s2
s3
Recurrent classes:
{s1,s2},{s3}
Transient classes:
{s4},{s5}
The Markov chain is not irreducible
The absorbing states are: s3

Recurrent classes: {s1,s2},{s3}, Transient classes: {s4}, {s5} and absorbing states: {s3}

a. Markov Chain in 2 (a) has the stationary distribution:

\[
\text{steadyStates}(\text{dtmcA1})
\]

\[
\begin{array}{ccc}
s1 & s2 & s3 \\
[1,] & 0.3333333 & 0.3333333 & 0.3333333
\end{array}
\]

b. Markov Chain in 2 (b) has the stationary distribution:

\[
\text{steadyStates}(\text{dtmcA2})
\]

\[
\begin{array}{cccc}
s1 & s2 & s3 & s4 \\
[1,] & 0.1666667 & 0.1666667 & 0.3333333 & 0.3333333
\end{array}
\]

c. Markov Chain in 2 (c) has no stationary distribution (since it is reducible and periodic).

It does have 2 (non-unique) sets of long run probabilities which you can see by calling the “steadyStates” function in the markovchain package.

d. Markov Chain in 2 (d) has no stationary distribution (since it is reducible and periodic).

It does have 2 (non-unique) sets of long run probabilities which you can see by calling the “steadyStates” function in the markovchain package.
6. Algorithm to simulate the Gambler’s ruin problem for general $p$ and $N$.

A gambler who at each play of the game has probability $p$ of winning and $1-p$ of losing. The gambler gains one unit if he wins otherwise loses one unit. Suppose that successive plays of the game are independent, starting with $i$ units, we let $X_n$ be the gambler’s fortune at time $n$. The algorithm for simulating the Gambler’s ruin problem for general $p$ and $N$ is given below:

```r
# Function to simulate the Gambler’s ruin problem.
# Requires the following input: n = number of time steps in sim,
# P = transition probability, x1 = initial state,
# N = largest value a gambler can earn.
MC.sim <- function(n,p, x1, N) {
  sim <- as.numeric(n)
  p1=p
  if (missing(x1)) {
    sim[1] <- 0  # random initial condition
  } else {
    sim[1] <- x1
  }
  for (i in 2:n)
  {
    if (x1==0){
      sim[i:n] <- 0
      break
    }
    else if (x1==N){
      sim[i:N] <- N
      break
    }
    else{
      sim[i] <- sample(c(-1,1),1,prob=c(1-p,p)) +x1
      x1=sim[i]
    }
    }
  sim
}

plot(NULL, xlim=c(0,50), ylim = c(0,10), ylab="State Space",
     xlab="Time", main = "Plot of Gambler ruin paths, p=0.4")
for (i in 1:5)
{
  X<-MC.sim(50,0.4,3,10)
}
a. Transition matrices $P$ and $P_T$

```r
p = 0.4
N = 10
x1 = 3
M = matrix(0, (N+1), (N+1))
M[1,1] = 1
M[(N+1), (N+1)] = 1
for (i in 2:(N))
{
  M[i,i+1] <- p
  M[i,i-1] <- 1-p
}

library(markovchain)

# Creating a discrete time Markov chain (DTMC) for Gambler ruin Problem
```
dtmcA <- new("markovchain", transitionMatrix=M, states=paste(letters[1:26])[1:(N+1)], name=ts1)
dtmcA

Gambler's Ruin Markov Chain
A 11 - dimensional discrete Markov Chain with following states
a b c d e f g h i j k
The transition matrix (by rows) is defined as follows
a b c d e f g h i j k
a 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
b 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
c 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0
d 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0
e 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0
f 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0
g 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0
h 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0
i 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0
j 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4
k 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
The transition probability matrix coresponding to transcient state ($P_T$) is given by
dtmcA[2:(N),2:(N)]

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. The matrix of mean time spent in transient states.

PT<-dtmcA[2:(N),2:(N)]
S<-solve(diag(1,nrow(PT),ncol(PT))-PT)
S

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>1.6519604 1.0866006 0.7096941 0.4584231 0.2909091 0.1792331 0.1047824</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1.6299009 2.7165015 1.7742352 1.1460577 0.7272727 0.4480827 0.2619561</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>1.5968117 2.6613529 3.3710470 2.1775097 1.3818182 0.8513572 0.4977165</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
c. The expected amount of time the gambler has $7, given that they started with $4.

\[ S[4,7] \]

\[ [1] \ 0.8513572 \]

d. The probability that starting with $3, the gambler reaches $9 before going broke is given by

\[ f_{3,9} = \frac{S_{3,9} - \delta_{3,9}}{S_{9,9}} \]

\[ (S[3,9]-0)/S[9,9] \]

\[ [1] \ 0.06342914 \]

e. BONUS: The probability that starting with $1, the gambler reaches $10 before going broke.

The general formula for this probability is

\[ P(X_n = N \mid X_0 = i) = \frac{1 - r^i}{1 - r^N} \]
where \( r = \frac{1 - p}{p} \). So for \( N = 10, i = 1, \) and \( p = 0.4, \) we have that \( r = \frac{0.6}{0.4} = 1.5 \) and

\[
P(X_n = N \mid X_0 = 1) = \frac{1 - 1.5^1}{1 - 1.5^{10}} = 0.00882.
\]