Homework 8

1. The following code will create a function to simulate a Poisson process

```
poissonsim=function(lambda=1,T ){
   x=c(0, 0) # set initial values to X(0)=0 and time=0
   dim(x)=c(2,1) #set dimensions of the above
   t=0 #set time to zero
   i=1 #keep track of number of events
   while (t<T){
    i=i+1 #add one event
    interarrival=rexp(1,rate=lambda) #simulate interarrival time
    nextstate=x[1,i-1]+1 #process increases by one
    t=t+interarrival #increase the current time
    x=cbind(x,c(nextstate,t)) #add the new X(t) and new time
   }
   x # return a matrix with 2 rows and number of columns corresponding to
     # number of events
}
```

   This can be called with `x=poissonsim(2,10)`. This will store a matrix of two rows. The first row is the value of the process, and the second row is the time when the event occurred. You may plot the process with

```
plot( x[2,], x[1,], type="s",xlim=c(0,10))
```

   (Note that you can obtain the values of the process using `x[1,]` and the times with `x[2,]`.

   Now note that you may generate a Poisson random variable with mean of 2 using the command

```
N=rpois(1,2)
```

and you can generate any number of uniform random variables using the `runif()` command. For instance, to generate 30 uniform random variables from 0 to 2 would be

```
 y=runif(30,0,2)
```

Use these facts to simulate a Poisson process using a method other than `poissonsim()`. To check whether two distributions are the same, one can use a qq plot. In R, this can be done with the command `qqplot(x,y)` if `x` and `y` contain the two samples you wish to compare. Check whether the process using this second method and the process from `poissonsim()` are the same. Use `lambda=2` and time up to `20`.

2. The following code will simulate a single server queue

```
ssqueue=function(lambda=1, mu=1, T){
x=c(0,0) #set the initial values--the first is X(0) the second is time.
dim(x)=c(2,1) #set the dimentsion
t=0 #set time to zero
i=1 #keep track of number of events
while (t<T){
    i=i+1 # add one event

    if (x[1,i-1]==0) #if current state is zero--can only move up
    {interarrival=rexp(1,rate=lambda) #simulate interarrival
nextstate=1}
    else { #current state is not zero
interarrival=rexp(1,rate=(lambda+mu)) #simulate interarrival
    if (rbinom(1,1,lambda/(lambda+mu))==1) # generates bernoulli random var
    #to decide up or down
    {nextstate=x[1,i-1]+1} #move up
    else
    {nextstate=x[1,i-1]-1} #move down
    }
    t=t+interarrival #update current time
    x=cbind(x,c(nextstate,t)) #add the new X(t) and new time
    }
   x # return a matrix with 2 rows and number of columns corresponding to
     # number of events
}
```

You may call the function using `x=ssqueue(1,2,20)` to generate a single server queue up to time 20 with $\lambda = 1$ and $\mu = 2$. You can use the same command as for the Poisson process to plot the path. Generate several plots including ones with λ greater than, less than, and equal to μ.

Also, verify that the mean of the stationary distribution is $\frac{1-\lambda/\mu}{\lambda/\mu}$ and variance is $\frac{1-\lambda/\mu}{(\lambda/\mu)^2}$ using simulation. To do this it may be helpful to use a for loop. To do this in R, use

```
for (i in 1:100){
insert commands here
}
```

3. Modify the code for a single server queue to simulate a birth/death process. Simulate examples with λ greater than, less than, or equal to μ. Show plots and discuss them in the context of the mean of this process given in class. (Note: To get reasonable results, you will need to reset the initial value for the process to something other than zero.)