1. The video for today presented a discrete-time queue simulation. At time 0, the queue contains one individual. In each time interval, X individuals enter the queue and Y individuals exit the queue, where both X and Y are Poisson random variables with mean 5.

   (a) Modify the code so that the queue has a maximum size of 100. That is, if 100 individuals are in the queue, no more may join until some leave.

R:

```r
# queue simulation as a function,
# with max queue size 100,
# stops when queue is empty
queueSimulation <- function(){
  queueSize <- 1
  time <- 0
  while(queueSize > 0){
    # individuals enter the queue
    x <- rpois(1, 5)
    if(queueSize + x > 100){
      queueSize <- 100
    } else {
      queueSize <- queueSize + x
    }
    # individuals leave the queue
    y <- rpois(1, 5)
    if(y > queueSize){
      queueSize <- 0
    } else {
      queueSize <- queueSize - y
    }
    # increment time
    time <- time + 1
  }
  return(time)
}

# estimate time at which queue is empty
times = replicate(1000, queueSimulation())
print(times)
print(mean(times))
```

Mathematica:

```
In[1]:= queueSim[] := Module[{},
          queueSize = 1;
          time = 0;
          While[queueSize > 0,
           time += 1;
           x = RandomVariate[PoissonDistribution[5]];
           queueSize = queueSize + x;
           If[queueSize > 100, queueSize = 100];
           y = RandomVariate[PoissonDistribution[5]];
           If[y > queueSize, queueSize = 0, queueSize = queueSize - y];
           (*Print["at time ",time," the queue contains: ",queueSize]*)
          ];
          (*Print["time until the queue is empty: ",time];*)
          Return[time]
         ]

In[2]:= queueSim[]

Out[2]= 17

In[3]:= vals = Table[queueSim[], 10000]

In[4]:= Mean[vals] // N

Out[4]= 52.3655

In[5]:= StandardDeviation[vals] // N

Out[5]= 245.383
```

(b) Let T be the first time at which the queue is empty. Estimate $E(T)$.

Averaging over many simulations, we find $E(T) \approx 52$.

(c) Let Z be the first time at which the size of the queue is first reaches 20. Estimate $E(Z)$.

Averaging over many simulations, we find $E(Z) \approx 54$.

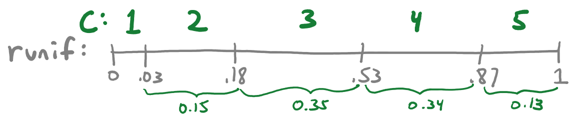
2. Suppose that C, the number of chips awarded in the game Plinko, has the following distribution:

c	1	2	3	4	5
$p(c)$	.03	.15	.35	.34	.13

What are two ways of simulating values of C in **R**?

1. Inverse CDF Method

Use runif(), then use inverse CDF to determine C.




```
chips <- NULL
for(i in 1:10000){
  u <- runif(1)
  if(u < 0.03){
    x <- 1
  } else if(u < 0.18){
    x <- 2
  } else if(u < 0.53){
    x <- 3
  } else if(u < 0.87){
    x <- 4
  } else{
    x <- 5
  }
  #print(x)
  chips[i] <- x
}
hist(chips)
print(c(mean(chips), sd(chips)))
```

2. Use sample()

sample(elements, size, replace = FALSE, prob = NULL)

vector of weights

```
nchips <- 1:5
pchips <- c(.03,.15,.35,.34,.13)
sample(nchips, 1, TRUE, pchips)
```

Use simulation to estimate the mean and standard deviation of C.

true mean: 3.39 SD: 0.978

3. Suppose that X, the winnings for one chip in Plinko, has the following distribution:

x	\$0	\$100	\$500	\$1000	\$10,000
$p(x)$	.39	.03	.11	.24	.23

Write a simulation of Plinko, taking into account both the number of chips a contestant earns and the amount of money won on each chip.

```
winnings <- NULL
nchips <- 1:5
pchips <- c(.03,.15,.35,.34,.13)
nwin <- c(0,100,500,1000,10000)
pwin <- c(.39,.03,.11,.24,.23)
for(i in 1:100){
  chips <- sample(nchips, 1, TRUE, pchips)  #number of chips the contestant earns
  amounts <- sample(nwin, chips, TRUE, pwin)  #amounts won from each chip
  winnings[i] <- sum(amounts)
}
hist(winnings)
```

What is the probability that a contestant wins more than \$11,000?

```
sum(winnings > 11000)
mean(winnings > 11000)
```

probability is about 0.3