DEMONSTRATIONS:

X:

x	1	2	3	4
$p(x)$	0.4	0.3	0.2	0.1

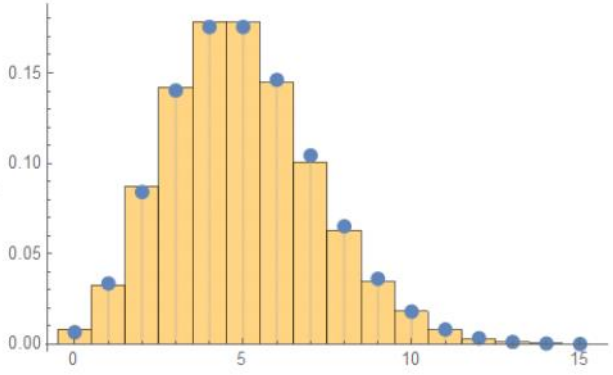
R: histogram of simulated values →

```r
randVar <- function(){
  u <- runif(1)
  if(u < 0.4){
    x <- 1
  } else if(u < 0.7){
    x <- 2
  } else if(u < 0.9){
    x <- 3
  } else {
    x <- 4
  }
  return(x)
}

randVar()

vals <- replicate(1000, randVar())

vals
hist(vals)
```

Mathematica: simulation of Poisson-distributed values:

```
randVal := Module[{u, x},
  u = RandomReal[];
  x = Which[u < 0.4, 1, u < 0.7, 2, u < 0.9, 3, True, 4];
  Return[x]
 ]

In[ ]:= vals = RandomVariate[PoissonDistribution[5], 10000];

In[ ]:= hist = Histogram[vals, Automatic, "Probability"]

In[ ]:= disc = DiscretePlot[Exp[-5] 5^x / x!, {x, 0, 15}, PlotMarkers → {Automatic, Medium}]

In[ ]:= Show[hist, disc]
```

Out[ ]=



Queue Simulations:

R:

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

# estimate time at which queue is empty
times = replicate(100000, queueSimulation())
times
mean(times)
sd(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
```