ANLY 511: Homework #4

Brody Vogel

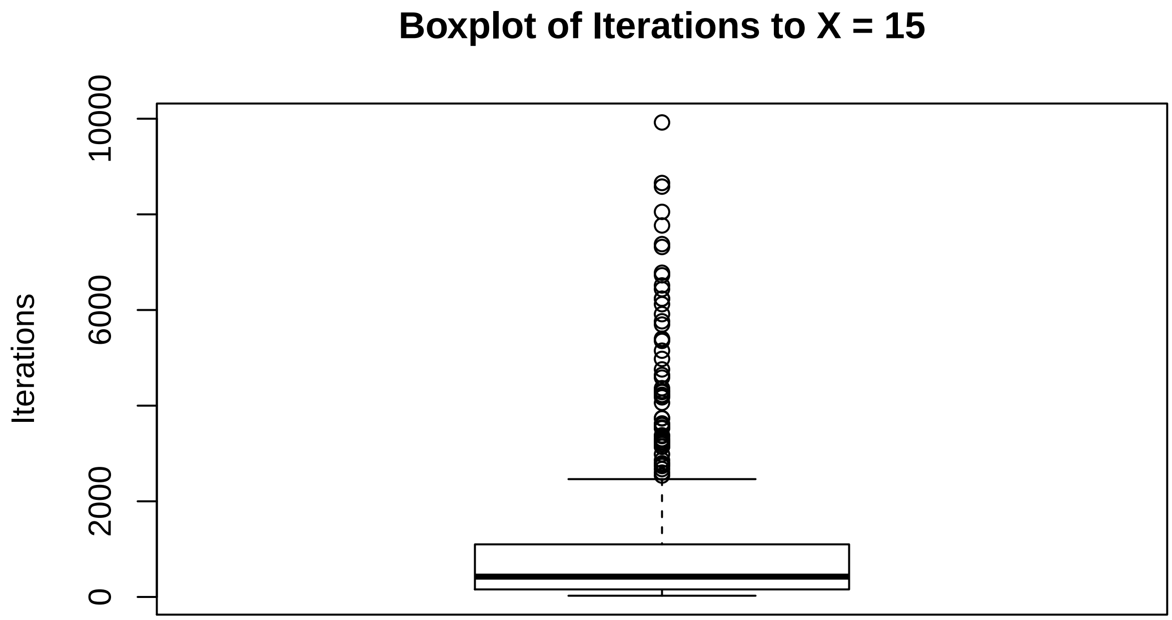
September 27th

Preparation

```
set.seed(1234)
dev.new(width = 1, height = 1)
```

Problem 25

```
### Random Walk Function
Walk <- function(){
  X <- 0
  iters <- 1
  while (X != 15 && iters < 10000){
    iters <- iters + 1
    y <- runif(1)
    if (y >= .5) {
      X <- X + 1 }
    else {
      X <- X - 1}
  }
  ### Because I'm using iters > 10000 as my cutoff for X never reaching 15
  if (iters == 10000) {
    iters <- NA
  }
  return(iters)
}
### Get the 500 samples and output summaries
Walks <- replicate(500, Walk())
boxplot(Walks, main = 'Boxplot of Iterations to X = 15', ylab = 'Iterations')
```




```r
summary(Walks)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##      26     159     425    1083    1099    9922      64
```

```r
### Compute the pseudo-number of walks that never hit 15
NA_Percent <- length(Walks[is.na(Walks) == TRUE])/500
NA_Percent
```

```
## [1] 0.128
```

In my simulation, the probability of never hitting X = 15 ≈ .12. Because the state space of X is finite, though, X will always eventually reach 15, if the number of trials is allowed to approach ∞.

Problem 26

The key aspects of the St. Petersburg system are to double down after a loss and bet a marginal value after a win. The idea is that, as long as I don't run out of money, I'll come out ahead with every win. In the proposed situation, in which I can choose a block of six numbers and the casino pays out six times my wager if one of those numbers comes up, I can mimic the St. Petersburg system in the following way:

```
After Win: bet marginal value
After Five Consecutive Losses: double down
  After Every Three More Consecutive Losses: double down again
```

This will ensure that I always expect to be ahead, given I can sustain enough consecutive losses.

As an example, a potential beginning to my using the original St. Petersburg system could be:

$50 \to 49 \to 47 \to 43 \to 35 \to (WIN)51$

And an example of my modified St. Petersburg system for the new situation could be:

$100 \to 99 \to 98 \to 97 \to 96 \to 95 \to 93 \to 91 \to 89 \to 85 \to 81 \to (WIN)105$

Both systems benefit from a larger starting value: I expect to be ahead if I can eventually win, and I can incur more consectuive losses - and so buy myself more time - with a larger beginning value.

Problem 27

Because 0 and 00 do not count as either odd or even: $S = -3, -1, +1, +3$

$Pr(X = -3) = Pr(\neg Black \cap \neg Even) = 12/38$

$Pr(X = -1) = Pr(Black \cap \neg Even) = 8/38$

$Pr(X = +1) = Pr(\neg Black \cap Even) = 8/38$

$Pr(X = +3) = Pr(Black \cap Even) = 10/38$

So $E[X] = -3(12/38) - 1(8/38) + 1(8/38) + 3(10/38) = -6/38 \approx$ -$.16.

Problem 28

Not quite. At some points during the walk, the frog has three options, while at others it has only two. To illustrate, if we consider the concrete example where the room is 2 x 10:

[1 2 3 4 5 6 7 8 9 10]

[11 12 13 14 15 16 17 18 19 20]

If the frog is at position 1, $Pr(X_{i+1} = 2|X_i = 1) = Pr(X_{i+1} = 11|X_i = 1) = 1/2$.

And if the frog is at position 2, $Pr(X_{i+1} = 1|X_i = 2) = Pr(X_{i+1} = 3|X_i = 2) = Pr(X_{i+1} = 12|X_i = 2) = 1/3$.

So $Pr(X_{i+1} = 2|X_i = 1) = 1/2 \neq 1/3 = Pr(X_{i+1} = 1|X_i = 2)$.

Thus, the transition matrix for this random walk cannot be symmetric.

Problem 29

$Pr(X = k) = 1/n$

$Pr(Y = i) = \binom{k}{i} p^i (1-p)^{k-i}$

So $Pr(X = k, Y = i) = Pr(Y = i|X = k) \times Pr(X = k) = \frac{\binom{k}{i} p^i (1-p)^{k-i}}{n}$

Problem 30

a) $Pr(X_3 = 3|X_0 = 1) = TM^3_{1,3} = .0625$, where TM = the transition matrix.

```
(transition_matrix%*%transition_matrix%*%transition_matrix)[1,3]
```

```
## [1] 0.0625
```

1) Since there are three possible ways to reach room 3 from room 1 in three iterations, $Pr(X_3 = 3|X_0 = 1) = [1/4 \times 1/2 \times 1/6] + [1/4 \times 1/6 \times 1/2][1/2 \times 1/4 \times 1/6] = .0625$

b) $1/2 < TM^T_{k,9}$, where TM is the transition matrix. A simulation shows that the caveman has less than a 50 percent chance of being alive after 87 trials, no matter where he begins his trek.

The second simulation shows that, after 86 steps, if the caveman starts in 2 of the 8 spots, he will have a > 50 percent chance of being alive.

```
### Initiate probabilities
steps <- 0
p1 <- 0
p2 <- 0
```

```
p3 <- 0
p4 <- 0
p5 <- 0
p6 <- 0
p7 <- 0
p8 <- 0
transition_matrix1 <- transition_matrix

### Loop until all ps are > .5
while (p1 <= .5 || p2 <= .5 || p3 <= .5 || p4 <= .5 || p6 <= .5 || p7 <= .5 || p8 <= .5) {
  steps <- steps + 1
  transition_matrix1 <- transition_matrix1 %*% transition_matrix
  p1 <- transition_matrix1[1,9]
  p2 <- transition_matrix1[2,9]
  p3 <- transition_matrix1[3,9]
  p4 <- transition_matrix1[4,9]
  p5 <- transition_matrix1[5,9]
  p6 <- transition_matrix1[6,9]
  p7 <- transition_matrix1[7,9]
  p8 <- transition_matrix1[8,9]
}
### Get results
steps
```

```
## [1] 87
```

```
transition_matrix1[,9]
```

```
## [1] 0.5426404 0.5143099 0.5001974 0.5877459 0.5268131 0.5044168 0.7164653
## [8] 0.8555842 1.0000000
```

```
### Show that one step less would result in at least one p being <= .5
transition_matrix11 <- transition_matrix
for (x in 1:86){
  transition_matrix11 <- transition_matrix11%*%transition_matrix}
transition_matrix11[,9]
```

```
## [1] 0.5384078 0.5098151 0.4955721 0.5839308 0.5224340 0.4998305 0.7138414
## [8] 0.8542477 1.0000000
```

Problem 31

a) $Pr(N = n) = e^{-\lambda}\frac{\lambda^n}{n!}$

$Pr(X = k) = \binom{n}{k}p^k(1-p)^{n-k}$

So $Pr(N = n, X = k) = Pr(X = k|N = n) \times Pr(N = n) = e^{-\lambda}\frac{\lambda^n}{n!} \times \binom{n}{k}p^k(1-p)^{n-k}$

From the simulations, it looks like $E[X] \approx \lambda p$. This makes sense, because E[N], N~Pois(λ) is λ and E[X], X~Bin(n,p) is np. So, if it's given that the number of binomial trials we have is equal to the value of a Poission-distributed random variable, we'd expect the outcome of this conditional distribution to be the expected number of trials, λ, multiplied by the likelihood of success, p: λp.

```
### Create k values for X
simulator <- function(lam, p, k) {
  num <- 0
```

```r
  nums <- c()
  while (num < k) {
    pois <- rpois(1, lam)
    bin <- rbinom(1, pois, p)
    num <- num + 1
    nums <- c(nums, bin)
  }
  return(nums)
}
### Test values of lambda and p
tester <- function(lambda, p) {
  point <- mean(replicate(10000, simulator(lambda, p, 1)))
  sprintf('For lambda = %f and p = %f, from 10000 simulations, E[X] = %f', lambda, p, point)
}
sapply(c(1,2,3,4,5,6,7,8,9,10, 50, 100), function(x)tester(x, .2))
```

```
##  [1] "For lambda = 1.000000 and p = 0.200000, from 10000 simulations, E[X] = 0.199000"
##  [2] "For lambda = 2.000000 and p = 0.200000, from 10000 simulations, E[X] = 0.399100"
##  [3] "For lambda = 3.000000 and p = 0.200000, from 10000 simulations, E[X] = 0.596700"
##  [4] "For lambda = 4.000000 and p = 0.200000, from 10000 simulations, E[X] = 0.784900"
##  [5] "For lambda = 5.000000 and p = 0.200000, from 10000 simulations, E[X] = 1.008100"
##  [6] "For lambda = 6.000000 and p = 0.200000, from 10000 simulations, E[X] = 1.195900"
##  [7] "For lambda = 7.000000 and p = 0.200000, from 10000 simulations, E[X] = 1.413000"
##  [8] "For lambda = 8.000000 and p = 0.200000, from 10000 simulations, E[X] = 1.619400"
##  [9] "For lambda = 9.000000 and p = 0.200000, from 10000 simulations, E[X] = 1.812500"
## [10] "For lambda = 10.000000 and p = 0.200000, from 10000 simulations, E[X] = 2.022900"
## [11] "For lambda = 50.000000 and p = 0.200000, from 10000 simulations, E[X] = 10.032900"
## [12] "For lambda = 100.000000 and p = 0.200000, from 10000 simulations, E[X] = 20.004000"
```

```r
sapply(c(1,2,3,4,5,6,7,8,9,10, 50, 100), function(x)tester(x, .4))
```

```
##  [1] "For lambda = 1.000000 and p = 0.400000, from 10000 simulations, E[X] = 0.407800"
##  [2] "For lambda = 2.000000 and p = 0.400000, from 10000 simulations, E[X] = 0.786000"
##  [3] "For lambda = 3.000000 and p = 0.400000, from 10000 simulations, E[X] = 1.188100"
##  [4] "For lambda = 4.000000 and p = 0.400000, from 10000 simulations, E[X] = 1.599500"
##  [5] "For lambda = 5.000000 and p = 0.400000, from 10000 simulations, E[X] = 1.995500"
##  [6] "For lambda = 6.000000 and p = 0.400000, from 10000 simulations, E[X] = 2.377400"
##  [7] "For lambda = 7.000000 and p = 0.400000, from 10000 simulations, E[X] = 2.811000"
##  [8] "For lambda = 8.000000 and p = 0.400000, from 10000 simulations, E[X] = 3.209300"
##  [9] "For lambda = 9.000000 and p = 0.400000, from 10000 simulations, E[X] = 3.587200"
## [10] "For lambda = 10.000000 and p = 0.400000, from 10000 simulations, E[X] = 3.979900"
## [11] "For lambda = 50.000000 and p = 0.400000, from 10000 simulations, E[X] = 19.934500"
## [12] "For lambda = 100.000000 and p = 0.400000, from 10000 simulations, E[X] = 40.081500"
```

```r
sapply(c(1,2,3,4,5,6,7,8,9,10, 50, 100), function(x)tester(x, .6))
```

```
##  [1] "For lambda = 1.000000 and p = 0.600000, from 10000 simulations, E[X] = 0.597000"
##  [2] "For lambda = 2.000000 and p = 0.600000, from 10000 simulations, E[X] = 1.202700"
##  [3] "For lambda = 3.000000 and p = 0.600000, from 10000 simulations, E[X] = 1.808400"
##  [4] "For lambda = 4.000000 and p = 0.600000, from 10000 simulations, E[X] = 2.398400"
##  [5] "For lambda = 5.000000 and p = 0.600000, from 10000 simulations, E[X] = 2.981400"
##  [6] "For lambda = 6.000000 and p = 0.600000, from 10000 simulations, E[X] = 3.593600"
##  [7] "For lambda = 7.000000 and p = 0.600000, from 10000 simulations, E[X] = 4.173600"
##  [8] "For lambda = 8.000000 and p = 0.600000, from 10000 simulations, E[X] = 4.788200"
##  [9] "For lambda = 9.000000 and p = 0.600000, from 10000 simulations, E[X] = 5.432600"
```

```
## [10] "For lambda = 10.000000 and p = 0.600000, from 10000 simulations, E[X] = 6.008200"
## [11] "For lambda = 50.000000 and p = 0.600000, from 10000 simulations, E[X] = 29.958200"
## [12] "For lambda = 100.000000 and p = 0.600000, from 10000 simulations, E[X] = 59.898900"
```

```
sapply(c(1,2,3,4,5,6,7,8,9,10, 50, 100), function(x)tester(x, .8))
```

```
##  [1] "For lambda = 1.000000 and p = 0.800000, from 10000 simulations, E[X] = 0.795400"
##  [2] "For lambda = 2.000000 and p = 0.800000, from 10000 simulations, E[X] = 1.611200"
##  [3] "For lambda = 3.000000 and p = 0.800000, from 10000 simulations, E[X] = 2.430800"
##  [4] "For lambda = 4.000000 and p = 0.800000, from 10000 simulations, E[X] = 3.173800"
##  [5] "For lambda = 5.000000 and p = 0.800000, from 10000 simulations, E[X] = 4.020100"
##  [6] "For lambda = 6.000000 and p = 0.800000, from 10000 simulations, E[X] = 4.783700"
##  [7] "For lambda = 7.000000 and p = 0.800000, from 10000 simulations, E[X] = 5.622100"
##  [8] "For lambda = 8.000000 and p = 0.800000, from 10000 simulations, E[X] = 6.400800"
##  [9] "For lambda = 9.000000 and p = 0.800000, from 10000 simulations, E[X] = 7.183000"
## [10] "For lambda = 10.000000 and p = 0.800000, from 10000 simulations, E[X] = 7.980000"
## [11] "For lambda = 50.000000 and p = 0.800000, from 10000 simulations, E[X] = 40.018200"
## [12] "For lambda = 100.000000 and p = 0.800000, from 10000 simulations, E[X] = 80.113200"
```

Problem 32

a) $Pr(X < 12|X < 18) = \frac{Pr(X \leq 11)}{Pr(X \leq 17)} \approx .15$

$E[X|X < 18] = \sum_{n=0}^{17} x \frac{Pr(X=x)}{Pr(X \leq 17)} \approx 14.0393$

```
### (a)
pbinom(11, 80, .2)/pbinom(17, 80, .2)
```

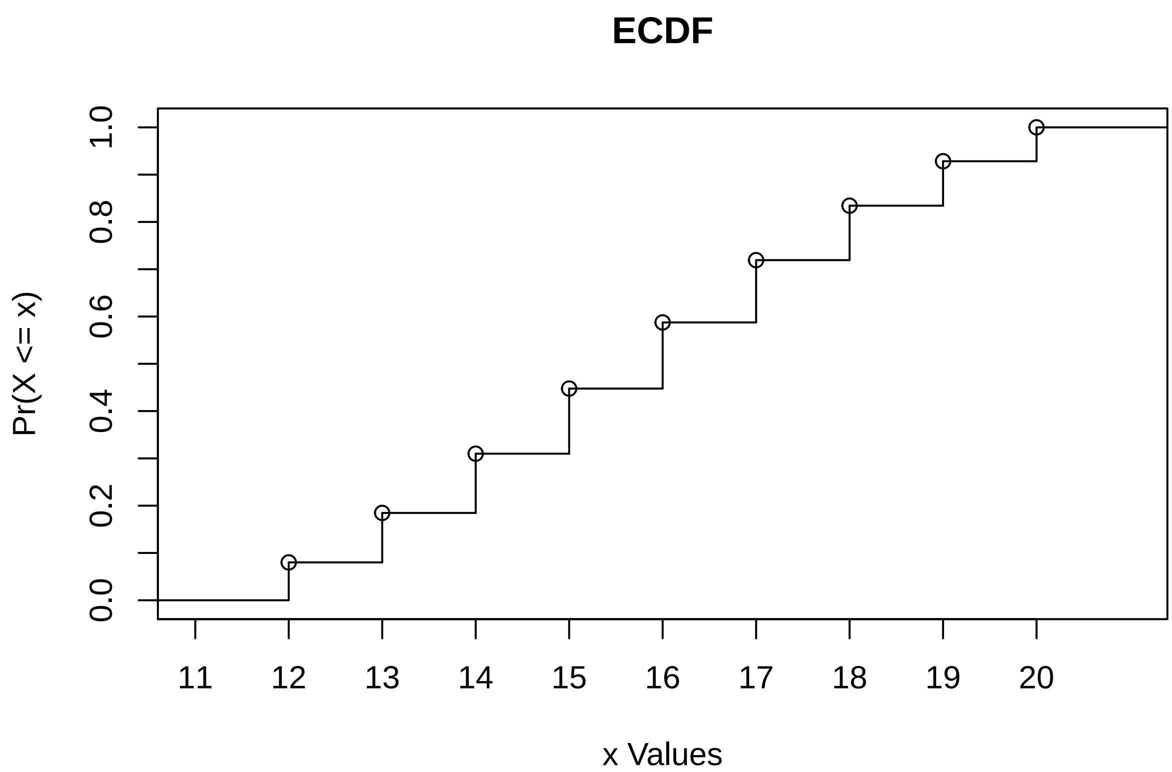
```
## [1] 0.1499783
```

```
sum <- 0
for (x in 0:17) {
  sum <- sum + x * (dbinom(x, 80, .2)/pbinom(17, 80, .2))
}
print(sum)
```

```
## [1] 14.0393
```

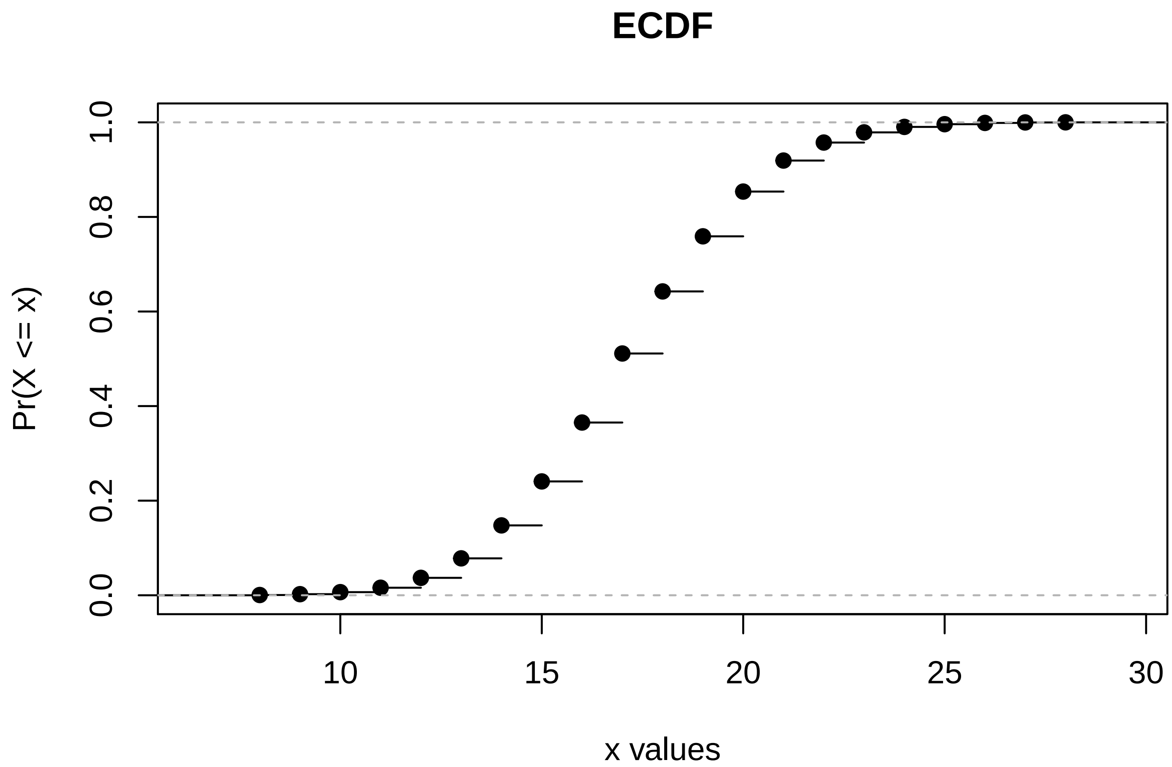
b)

```
vecs <- c()
for (x in 12:20) {
  vecs <- c(vecs, dbinom(x, 80, .2)/(pbinom(20, 80, .2) - pbinom(11, 80, .2)))
}
newVecs <- c()
sum <- 0
for (vec in vecs) {
  sum <- sum + vec
  newVecs <- c(newVecs, sum)
}
newVecs <- c(0, newVecs)
sfun <- stepfun(12:20, newVecs)
plot(sfun, xaxt = 'n', yaxt = 'n', xlab = 'x Values', ylab = 'Pr(X <= x)', main = 'ECDF')
axis(1, at = 1:20)
axis(2, at = seq(0, 1, .1))
```



c)

```
Xs <- c()
while (length(Xs) < 10000) {
  X <- rbinom(1, 80, .2)
  Y <- rbinom(1, 100, .7)
  if (X + Y == 90) {
    Xs <- c(Xs, X)}
  else {
    Xs <- Xs}
}
plot(ecdf(Xs), main = 'ECDF', xlab = 'x values', ylab = 'Pr(X <= x)')
```



d)

$E[Z|X = 110] = DNE$, because X can never be > 80.

$E[Z|X = 15] \approx 85$

$E[Z|X = 20] \approx 90$

These results make sense, becasue we'd anticipate that our expected value would be the given value of X + E[Y], which is the case in the two following simulations.

```
helper <- function(x) {
  Zs <- c()
  while (length(Zs) < 10000) {
    X <- rbinom(1, 80, .2)
    if (X == x) {
      Zs <- c(Zs, X + rbinom(1, 100, .7))}
    else {
      Zs <- Zs}
  }
  return(Zs)
}
mean(helper(15))
```

```
## [1] 85.0076
```

```
mean(helper(20))
```

```
## [1] 89.958
```

e) $E[X|Z = 80] \approx 13.732$ $E[X|Z = 90] \approx 17.337$ $E[X|Z = 100] \approx 21.769$

These results also make sense, as we'd expect the values of X and Y that add to the desired Zs to split the desired Zs' differences from E[Z] according to their size. So, for example, when Z = 80, we'd expect X to be ≈ E[X] - 2.5 = 16 - 2.5 = 13.5 and Y to be ≈ E[Y] - 3.5 = 70 - 3.5 = 66.5, because the Z value of 80 is 6 from E[Z] = 86.

```r
helper1 <- function(z){
  Xs <- c()
  while (length(Xs) < 1000) {
    X <- rbinom(1, 80, .2)
    Y <- rbinom(1, 100, .7)
    if (X + Y == z) {
      Xs <- c(Xs, X)}
    else {
      Xs <- Xs}
  }
  return(Xs)
}

mean(helper1(80))
```

```
## [1] 13.705
```

```r
mean(helper1(90))
```

```
## [1] 17.395
```

```r
mean(helper1(100))
```

```
## [1] 21.814
```