Mini Project 1

Name: Tejas Ravi Rao (txr171830)

Section - 1

Q1(a) $\int_{15}^{\infty} f(t)dt = \int_{15}^{\infty} 0.2\exp(-0.1\mathrm{t}) - 0.2\exp(-0.2\mathrm{t})\,\mathrm{dt}$

$= \int_{15}^{\infty} 0.2\exp(-0.1\mathrm{t}) - \int_{15}^{\infty} 0.2\exp(-0.2\mathrm{t})\mathrm{dt}$

$= -0.2\int_{15}^{\infty} \exp(-t/10)\,dt - 0.2\int_{15}^{\infty} \exp(-t/5)\,dt$

$= -2\exp\left(-\frac{t}{10}\right) + \exp\left(-\frac{t}{5}\right) + C$

After applying the upper and lower limits $\int_{15}^{\infty} f(t)dt$

$= \frac{\exp(-3)\left(10\exp\left(\frac{3}{2}\right)-5\right)}{5}$

= 0.39647

Q1(b) The steps to take a Monte Carlo approach to compute E(T) and P(T > 15) is as follows:

(i) In order to simulate one draw of the block lifetimes Xa and Xb, I have used the rexp() function where the rate parameter is equal to 1/10 (as we have been given the lifetime to be 10 years for each block) which is 0.1. The code is as follows:

```
# Lifetime of Block A
# Simulate one draw of Xa
BlockA = rexp(1, rate = 0.1)

# Lifetime of Block B
# Simulate one draw of Xb
BlockB = rexp(1, rate = 0.1)
```

It is mentioned in the question that the satellite works until both blocks fail. Therefore, for the satellite lifetime T , we can choose the maximum of the two draws obtained above.

```
# Simulate one draw of satellite lifetime T
# Using max as SatelliteT lasts until both fail
# so longer lifetime considered
SatelliteT = max(BlockA, BlockB)
```

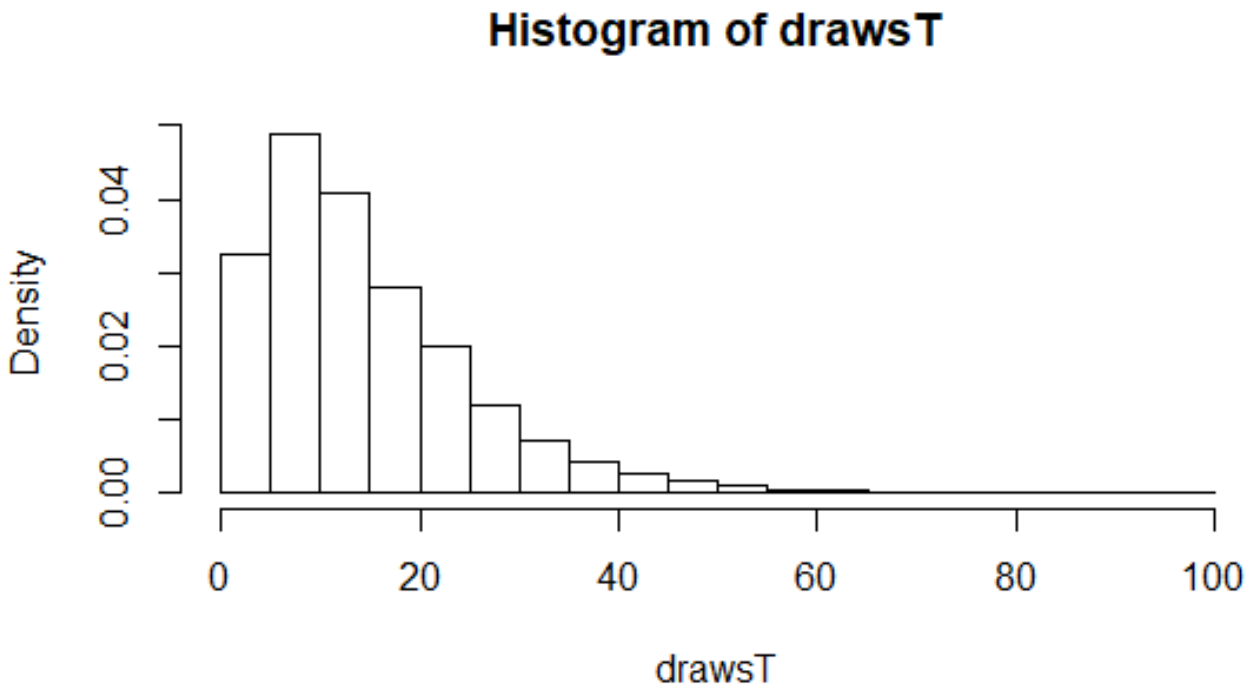
(ii) Repeat the previous step 10,000 times. This would give us 10,000 draws from distribution of T. As mentioned, the 'replicate' function was used to simulate 10,000 draws and saved in drawsT.

```
# perform step 10000 times to give
# 10000 distributions of T
 drawsT = replicate(10000, max(rexp(1,rate = 0.1), rexp(1,rate = 0.1)))
```

Note:
As asked in question, the above line of code is a single line of code for both steps (i) and (ii).

(iii) Made use of 'hist' function to get histogram of draws of T. Also, the freq parameter was set to FALSE to get density plot.

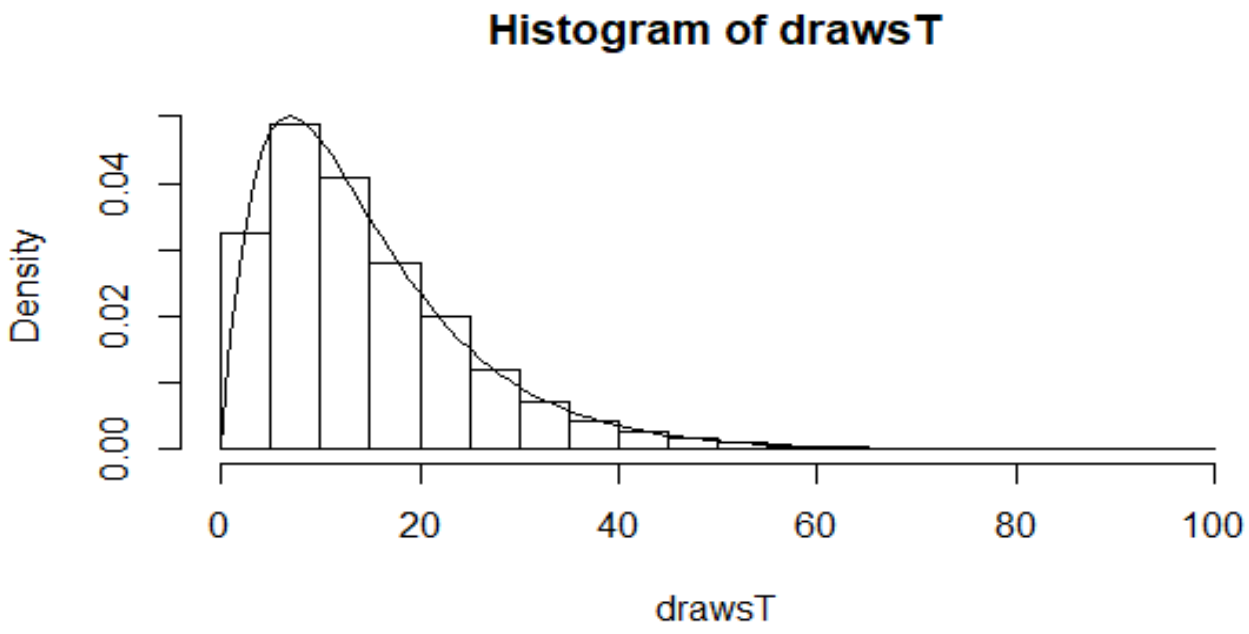
```
# histogram of draws of T
hist(drawsT, freq = FALSE)
```



Superimposed the density function given above. Made use of the 'curve' function for drawing density. The curve() function takes an expression as an input which would be our above-mentioned density function. Additionally, the *add* parameter in the curve() function is set to TRUE because we want to add this plot to already existing histogram. I have also set *xlab* and *ylab* label parameters.

```
# superimpose density function
curve((0.2*exp(-0.1*x)-0.2*exp(-0.2*x)), add = TRUE, xlab = "drawsT", ylab = "density value")
```

The plot is shown below:



(iv) Made used of the saved draws to estimate *E(T).* We make use of the mean() function. The following value was obtained.

 # find Expected value E(T)
mean(drawsT)

[1] 15.05923

The above obtained value is close to the value of *E(T)* = 15 mentioned in our question.

(v) Also, used the saved draws to estimate the probability that the satellite lasts more than 15 years. I made use of the condition to get values of drawsT greater than 15. The following value was obtained.

 # find probability that satellite lasts more than 15 years
mean(15 < abs(drawsT))

[1] 0.3957

Here, we notice that the probability value obtained above is almost close to the value found through the integral which is 0.39647.

(vi) Repeated the above process of finding E(T) and an estimate of the probability 4 more times. So, for this I made use of a helper function called 'parametersHelper'. This function takes as input the

number of Monte Carlo replications. By using the 'replicate' function on this helper function, I was able to repeat the above procedure 4 times.

```
# utility method to repeat above tasks using replicate
# reps is the no. of Monte Carlo repetitions
parametersHelper = function(reps){
  drawsT = replicate(reps, max(rexp(1,rate = 0.1), rexp(1,rate = 0.1)))
  ExpectedVal = mean(drawsT)
  Prob = mean(15 < abs(drawsT))
  print(c(ExpectedVal,Prob))
}
```

I replicate the above method 4 times with 10,000 Monte Carlo replications.

```
# replicate the operation 4 more times
replicate(4, parametersHelper(10000))
```

Therefore, the following values were noted:

Trial	E(T)	Probability
1	15.05923	0.39570
2	14.96067	0.39540
3	14.99596	0.39680
4	14.95351	0.39960
5	14.99027	0.39230

Q1(c)

Repeated part(vi) above 5 times using 1000 and 100,000 Monte Carlo replications instead of 10,000. So, similarly for this step I made used of my helper method 'parametersHelper' to perform the operation 5 times. The code is as follows.

```
# replicate the procedure 5 more times
# but using 1000 monte carlo replications
replicate(5, parametersHelper(1000))
```

Therefore, the following values were noted:

Trial	E(T)	Probability
1	14.68458	0.39700
2	14.62216	0.39300
3	15.38798	0.40600
4	14.97437	0.38800
5	15.56289	0.41800

Similarly

 # replicate the procedure 5 more times
 # but using 100000 monte carlo replications
replicate(5, parametersHelper(100000))

Therefore, the following values were noted:

Trial	E(T)	Probability
1	15.02149	0.39644
2	15.01205	0.39748
3	14.99578	0.39648
4	15.04372	0.39681
5	15.05177	0.39784

From each of the tables for 1,000 , 10,000 , and 100,000 repetitions, we can see that the values noted for 1000 repetitions are off and not close to the value obtained analytically. Therefore, when we check for 10,000 repetitions, we find the values approaching closer to the analytical values. Finally, for 100,000 repetitions we find the values for all 5 trial runs to be very close to given E(T) and analytical probability value.

This is because, if we perform the same calculation , the average of the results obtained from large number of trials should be close to the expected values E(T) and will tend to become closer as more trials are performed.

Q2)

A Monte Carlo approach to estimate the value of π based on 10, 000 replications. The following steps were taken.

(i) Firstly, let us consider a circle of radius r = 0.5, centered at point (0.5,0.5). This circle is inscribed in a square with coordinates – (0,0), (0,1), (1,0), (1,1). We can get a relation between π and the probability that a randomly chosen point in the square falls inside the circle.

(ii) Therefore, if we can estimate the probability, we can estimate the value of π. This can be done using Monte Carlo simulation by randomly generating many points (about 10,000) inside the square and finding the ratio of points that fall inside the circle.

(iii) The number of points within the circle to within the square will be almost equal to ratio of the area of circle to area of square.

(iv) To get the relation if we divide

$$\frac{Area\ of\ Circle}{Area\ of\ Square} = \frac{\pi(0.5)(0.5)}{(1)(1)}$$

$$\frac{Area\ of\ Circle}{Area\ of\ Square} = \frac{\pi}{4}$$

$$\pi = \frac{4 * \ Area\ of\ Circle}{Area\ of\ Square}$$

(v) Generate 10,000 X coordinates independently from a Uniform distribution. We make use of runif() function to generate the x values.

```
# generate 10000 X coordinates
xVal = runif(10000,0,1)
```

Similarly generate 10,000 Y coordinates

```
# generate 10000 Y coordinates
yVal = runif(10000,0,1)
```

(vi) Calculate distance between the points and the center of the circle and save the distance values.

```
# calculate distance between center of circle and all points
# center of circle at (0.5,0.5)
distVal = sqrt((xVal-0.5)^2 + (yVal-0.5)^2)
```

(vii) Find the total number of points that lie within circle.

```
# Find the total number of points within circle
circlePoints = length(which(distVal <= 0.5))
print(circlePoints)
```

Find the total number of points that lie within square.

```
# Find the total number of points within square
squarePoints = length(distVal)
print(squarePoints)
```

(viii) Calculate ratio and calculate Pi

```
# calculate ratio of points within circle to within square
result = circlePoints/squarePoints
print(result)
# calculate value of pi from relation
piVal = result * 4
print(piVal)
```

Value of π obtained from relation = 3.1452

Section – 2

R Code

Q1)

```
# Random number generator seed
# Delivers uniform random variables
set.seed(11)

# Q1 b(i)
# Lifetime of Block A
# Simulate one draw of Xa
BlockA = rexp(1, rate = 0.1)

# Lifetime of Block B
# Simulate one draw of Xb
BlockB = rexp(1, rate = 0.1)

# Simulate one draw of satellite lifetime T
# Using max as SatelliteT lasts until both fail
# so longer lifetime considered
SatelliteT = max(BlockA, BlockB)

# Q1 b(ii)
# perform step 10000 times to give
# 10000 distributions of T
 drawsT = replicate(10000, max(rexp(1,rate = 0.1), rexp(1,rate = 0.1)))

# Q1 b(iii)
# histogram of draws of T
hist(drawsT, freq = FALSE)

# superimpose density function
curve((0.2*exp(-0.1*x)-0.2*exp(-0.2*x)), add = TRUE, xlab = "drawsT", ylab = "density value")

# Q1 b(iv)
# find Expected value E(T)
mean(drawsT)
```

```
# Q1 b(v)
# find probability that satellite lasts more than 15 years
mean(15 < abs(drawsT))

# utility method to repeat above tasks using replicate
# reps is the no. of Monte Carlo repetitions
parametersHelper = function(reps){
  drawsT = replicate(reps, max(rexp(1,rate = 0.1), rexp(1,rate = 0.1)))
  ExpectedVal = mean(drawsT)
  Prob = mean(15 < abs(drawsT))
  print(c(ExpectedVal,Prob))
}

# Q1 b(vi)
# replicate the operation 4 more times
replicate(4, parametersHelper(10000))

# Q1 c
# replicate the procedure 5 more times
# but using 1000 Monte Carlo replications
replicate(5, parametersHelper(1000))

# Q1 c
# replicate the procedure 5 more times
# but using 100000 Monte Carlo replications
replicate(5, parametersHelper(100000))
```

Q2)

```
# generate 10000 X coordinates
xVal = runif(10000,0,1)

# generate 10000 Y coordinates
yVal = runif(10000,0,1)

# calculate distance between center of circle and all points
# center of circle at (0.5,0.5)
distVal = sqrt((xVal-0.5)^2 + (yVal-0.5)^2)
```

```
# Find the total number of points within circle
circlePoints = length(which(distVal <= 0.5))
print(circlePoints)

# Find the total number of points within square
squarePoints = length(distVal)
print(squarePoints)

# calculate ratio of points within circle to within square
result = circlePoints/squarePoints
print(result)

# calculate value of pi from relation
piVal = result * 4
print(piVal)
```