STAT 2150 Assignment 1

Due Thursday, September 26 at 11:59 PM

Adejare Taiwo

Instructions:

- Knit this file to pdf to see the questions in a more readable format.
- Enter your name in Line 4.
- Do not change the code in the provided code chunks or in the questions. Simply fill in your answers in the code chunks that are missing code and answer the questions with text responses. Be sure when adding in text responses to never copy-paste symbols from outside of the document.
- Do not use any functions or methods that we have not learned in class.
- It is recommended that you knit to pdf after you fill in each code chunk.
- Your knit pdf file should show the result answering the question. To do this, after creating an R object, you should also print it in a new line within the code chunk, unless otherwise instructed.

Question 1:

The `data` vector below consists of 82 values.

```
data = c(59.13, 46.80, 53.49, 56.59, 53.94, 55.31, 47.35, 42.35, 54.01, 53.87,
         51.81, 59.21, 58.98, 54.72, 53.66, 54.87, 54.58, 47.10, 43.03, 54.38,
         65.70, 74.91, 82.11, 65.61, 53.89, 63.76, 47.05, 54.19, 57.91, 43.78,
         59.97, 48.77, 47.57, 83.04, 53.25, 41.31, 67.92, 60.18, 60.33, 55.03,
         53.20, 47.06, 66.63, 58.98, 59.74, 50.37, 50.43, 53.34, 78.11, 51.40,
         43.78, 56.11, 51.82, 58.98, 47.31, 77.36, 54.36, 52.04, 48.92, 47.54,
         48.95, 49.42, 55.80, 52.55, 58.17, 52.07, 43.71, 46.52, 58.51, 72.01,
         59.11, 47.56, 53.55, 74.86, 63.31, 61.97, 59.03, 67.03, 55.50, 42.12,
         47.90, 64.11)
```

Recall from your previous courses that we can identify outliers in the dataset by locating any value that is either:

- higher than the "upper fence", which is calculated as UF = Q3 + 1.5*(Q3 - Q1)
- lower than the "lower fence", which is calculated as LF = Q1 - 1.5*(Q3 - Q1)

where Q1 is the 25th percentile and Q3 is the 75th percentile. Also, recall that the `fivenum()` function gives you the five-number summary for a dataset: minimum, Q1, median, Q3, maximum.

```
fiveNum = fivenum(data)
```

Write the R code that identifies which data values in the above dataset are outliers.

```r
Q1 = fiveNum[2]
Q3 = fiveNum[4]

IQR = Q3 - Q1

minVal = Q1 - IQR * 1.5
maxVal = Q3 + IQR * 1.5

outliers = data[(data < minVal) | (data > maxVal)]
outliers
```

```
## [1] 74.91 82.11 83.04 78.11 77.36 74.86
```

Question 2:

Write a function that takes a numeric vector as input and returns a vector of the same length with `TRUE` in components where the input vector has an even value and `FALSE` in components where the input vector has an odd value.

Hint: we don't need to write code that individually checks each value in the input vector. Instead, the whole vector can be checked for even and odd values all at once.

Show that your function gives the correct answer for each of the following two input vectors:

```
vec1 = c(7,7,5,2,6,4,8,3,1,7)
vec2 = c(3,5,5,3,2,9,6,5,2,1)
```

```
checkEven = function(nums) {
  isEven = (nums %% 2 == 0)
  return(isEven)
}

checkEven(vec1)
```

```
##  [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
checkEven(vec2)
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

Question 3:

Suppose you are calculating the overall performance of a basketball player based on four key statistics: Points Per Game (PPG), Assists Per Game (APG), Rebounds Per Game (RPG), and Steals Per Game (SPG). The player's overall performance is weighted as follows:

- PPG contributes 50%,
- APG contributes 20%,
- RPG contributes 20%,
- SPG contributes 10% to the overall performance.

Write a function that takes in a **vector x** of four values representing these statistics in the following format: (PPG, APG, RPG, SPG). The function should return a **list** (a type of R object) containing the following two elements:

1. The player's overall performance rating based on the weights.
2. A performance category based on the rating:
    - "Excellent" for ratings 20 or higher,
    - "Good" for ratings greater than or equal to 15 and less than 20,
    - "Average" for ratings greater than or equal to 10 and less than 15,
    - "Needs Improvement" for ratings below 10.

Show that when you call your function with the following stats, it returns the expected output:

- If a player has the following stats: PPG = 25, APG = 5, RPG = 10, and SPG = 2, the function should return a list with:

  - 15.7 as the overall performance
  - "Good" as the performance category

- If a player has the following stats: PPG = 35, APG = 8, RPG = 12, and SPG = 5, the function should return a list with:
  - 22 as the overall performance
  - "Excellent" as the performance category

```
calcResult = function (x) {
  PPG = 50
  APG = 20
  RPG = 20
  SPG = 10
  total = 100

  weights = c((PPG / total), (APG / total), (RPG / total), (SPG / total))

  score = sum(x * weights);

  if(score >= 20) {
    performance = "Excellent"
  } else if (score >= 15) {
    performance = "Good"
  } else if (score >= 10) {
    performance = "Average"
  } else {
    performance = "Needs Imporvement"
  }

  return(list(score, performance))
}

calcResult(c(25, 5, 10, 2))
```

```
## [[1]]
## [1] 15.7
##
## [[2]]
## [1] "Good"
```

```
calcResult(c(35, 8, 12, 5))
```

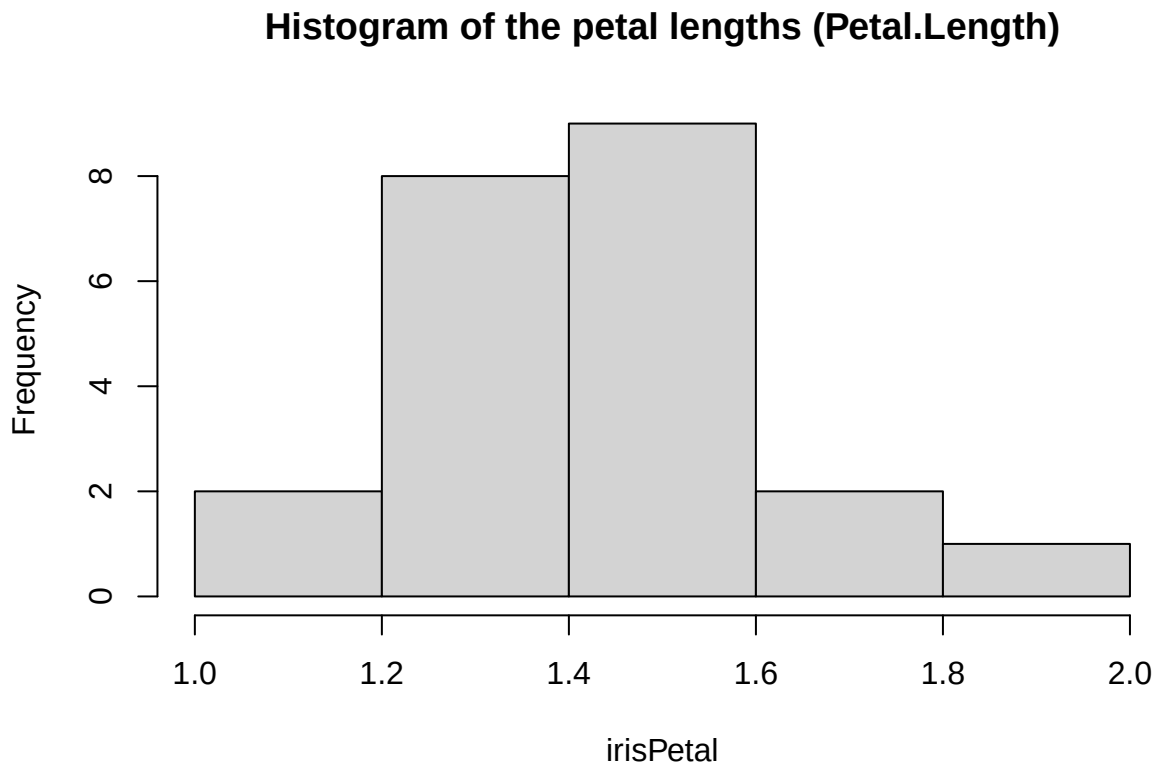
```
## [[1]]
## [1] 22
##
## [[2]]
## [1] "Excellent"
```

Question 4:

The **iris** dataset is built-in to R. See documentation on the dataset by typing `?iris` at the R console.

(a) Write R code that makes a histogram of the petal lengths (Petal.Length) for flowers of the species **setosa** where the sepal width (Sepal.Width) is at least 3.5.

```
irisPetal <- iris$Petal.Length[which(iris$Sepal.Width >= 3.5 & iris$Species == "setosa")]
hist(irisPetal, main = "Histogram of the petal lengths (Petal.Length)")
```



(b) Write R code that obtains the average petal length for flowers of the species **virginica** where the sepal width is at least 3.0 and petal length is greater than 5.0.

```
averagePetalLength = mean(iris$Petal.Length[which(iris$Species == "virginica" & iris$Sepal.Width >= 3.0
print(averagePetalLength)
```

```
## [1] 5.641379
```

(c) Create a new data frame that contains only flowers of the species **versicolor** that have a speal length of 5.0 or smaller. Sort this new data frame so that the sepal widths are in order from smallest to largest. Print the sorted data frame.

```r
versicolordata = iris[(iris$Species == "versicolor") & (iris$Sepal.Length <= 5.0),]
ordered = versicolordata[order(versicolordata$Sepal.Width),]
print(ordered)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 61          5.0         2.0          3.5           1 versicolor
## 94          5.0         2.3          3.3           1 versicolor
## 58          4.9         2.4          3.3           1 versicolor
```

(d) From the overall **iris** dataset, consider only flowers where the sepal length is 5.5 or greater. Use the `table()` function to determine the number of such flowers that are of each of the three species.

```r
irisFlower = iris[iris$Sepal.Length >= 5.5,]
flowerTable = table(irisFlower$Species)
print(flowerTable)
```

```
##
##     setosa versicolor  virginica
##          5         44         49
```

Question 5:

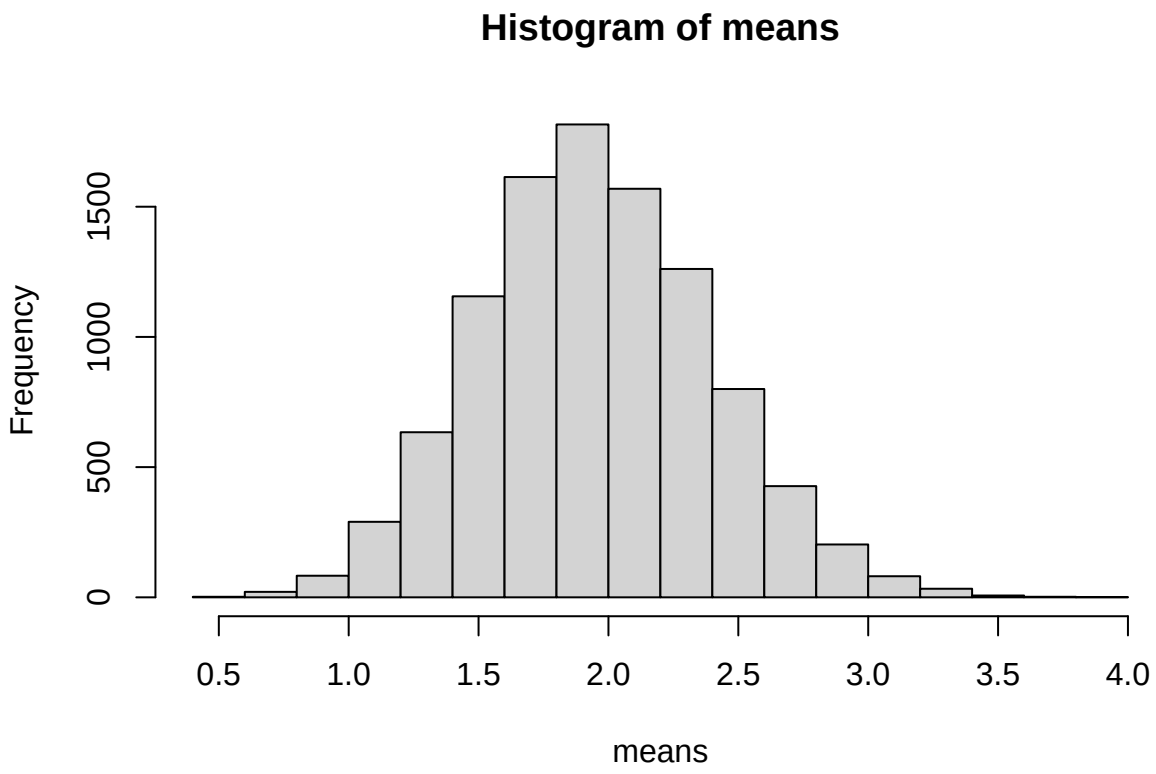
In the below code chunk, do not change any of the code except for inputting your student number in place of 1111111.

```r
set.seed(7980132) # change 1111111 to your 7-digit student number
lambda = sample(1:4,1)
x = rpois(1e5,lambda)
mymat = matrix(x,ncol=10)
means = apply(mymat,1,mean)
```

The code above generates 100,000 random values, where each value is a whole number between 0 and some upper bound (which will vary from student to student). The 100,000 values are placed in a matrix with 10,000 rows and 10 columns. For each row, we obtain the mean and the 10,000 means are in the `means` vector.

Make a histogram of the `means` vector.

```r
hist(means)
```



Calculate what proportion of the values in the **means** vector are in the range [1.55,2.45). (Knit this file to pdf to make this readable.)

```r
queryPortion = means[means >= round(lambda - sqrt(lambda/10),2) & means <= round(lambda + sqrt(lambda/1
portion = length(queryPortion) / length(means)
print(portion)
```

```
## [1] 0.6904
```

Calculate what proportion of the values in the **means** vector are less than 1.11. (Knit this file to pdf to make this readable.)

```r
newQueryPortion = means[means < round(lambda - 2*sqrt(lambda/10),2)]
newportion = length(newQueryPortion) / length(means)
print(newportion)
```

```
## [1] 0.0218
```