0.1 Question 2 :

A hardware store receives a shipment of 10,000 bolts that are supposed to be 12 cm long. The mean of this shipment of 10,000 bolts is indeed 12 cm, and the standard deviation is 0.2 cm.

For the following questions, determine if you have enough information to answer. If you do, then show all steps calculating the answer in the Markdown cell directly below. If you don't, explain what additional information you would need. If you use any theorems, cite the theorem and specify which assumptions are necessary for the theorem to hold.

Question 2a). What is the probability that a randomly chosen bolt is less than 10 cm long?

Question 2b). For quality control, the hardware store chooses 100 bolts at random to measure. They will declare the shipment defective and return it to the manufacturer if the average length of 100 bolts is less than 11.97 cm or greater than 12.04 cm. Find the probability that the shipment is found satisfactory (i.e. not defective).

0.1.1 Part a

We aren't told anything about what type of distribution this is. If we are going to calculate the probability that a randomly chosen bolt is less than 10 cm long, we would essentially be calculating the area under the curve and this is not possible if we do not know what type of distribution is present. Therefore, we cannot actually calculate this probability because **we need more information about the distribution**.

0.2 Part b

To answer this question, we need to use the Central Limit Theorem. We first need to calculate the standard error with

$$\sigma_x = \frac{\sigma}{\sqrt{n}}$$

where when we plug in the values in this example, $\sigma = 0.2$ and $n = 100$ we get

$$\sigma_x = \frac{0.2}{\sqrt{100}} = \frac{0.2}{10} = 0.2.$$

We then calculate the z score of each limit from our hardware store using

$$z = \frac{\bar{x}_x - \mu}{\sigma_x}$$

and we find

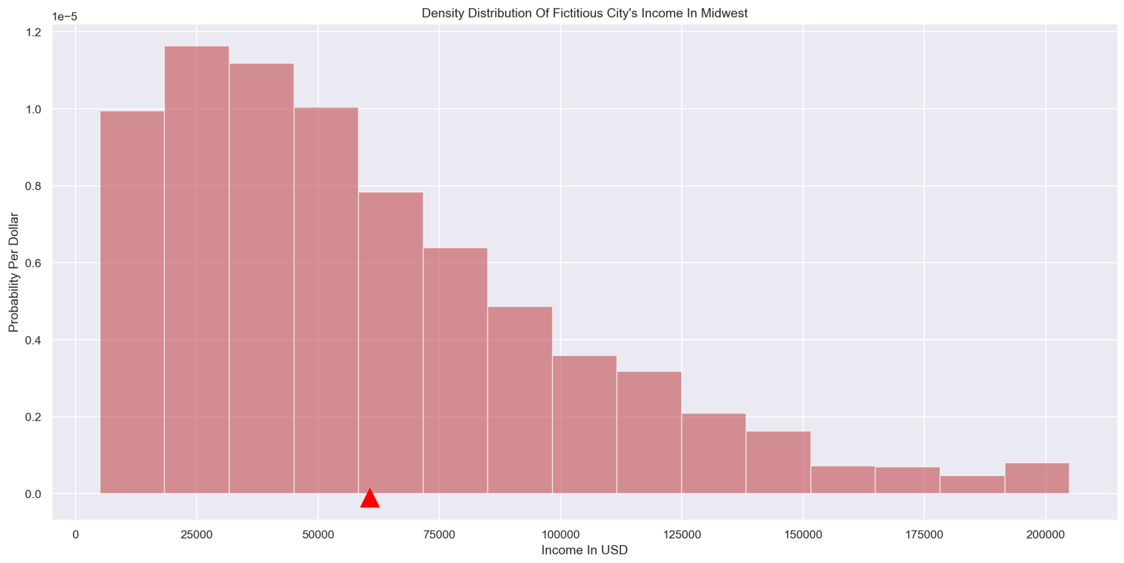
$$z_{11.97} = \frac{11.97 - 12}{0.02} = -1.5 \tag{1}$$
$$z_{12.04} = \frac{12.04 - 12}{0.02} = 2. \tag{2}$$

Because we cannot solve the CDF analytically, we have to use Python to do so. In Python, we would calculate this probability with `stats.norm.cdf(12.04) - stats.norm.cdf(11.97)`. This gives us a rough probability of 91.04%.

QUESTION 3A: Load the data into a pandas DataFrame called `dfIncome`, calculate the population mean of Income and then make a **density histogram** of the Distribution of the Income data **with 15 bins**. Include a title for your plot and label the x-axis (we have provided a label for the y-axis). Note we have included code to mark where the population mean lies on the histogram.

```
In [152]: dfIncome = pd.read_csv("income_data.csv")

          mean_income = dfIncome['Income'].mean()

          print("Population income mean is", mean_income)

          plt.hist(dfIncome['Income'], bins=15, density=True, alpha=0.6, color='r')
          # write code to plot density histogram above this line
          plt.xlabel("Income In USD")
          plt.ylabel("Probability Per Dollar") # Since this is a density histogram, the y-units are pro
          plt.title("Density Distribution Of Fictitious City's Income In Midwest")
          #Add a triangle marker to indicate where the population mean is
          plt.scatter(mean_income, -.0000001, marker='^', color='red', s=300);
```

```
Population income mean is 60613.8492
```



QUESTION 3B: Describe the shape of the Income distribution (i.e. comment on modality and skew)

The income distribution in this fictitious city happens to be **right skewed**. In terms of the **modality** of this distribution, the mode of the incomes in this city is $25,000. We can see from the distribution that this distribution is **unimodal** and the common income range is around the aforementioned mode.

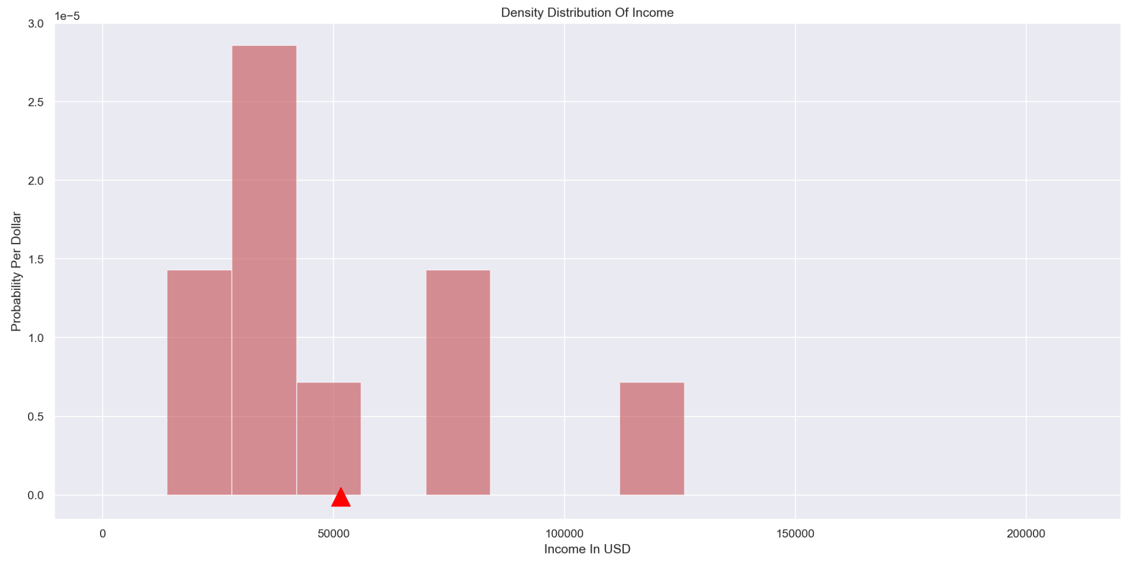
QUESTION 3C:

i). Write a function to collect a random sample of size `sample_size` with replacement from `dfIncome` and plot the **density histogram** of the empirical distribution of the income for your sample. Use 15 bins for your histogram and set the x-axis range to be from (0,210000). (Hint: use the dataframe method `.sample()`). Include a title and label for both axes.
Then run the cells provided below to output 3 separate distributions for sample sizes of 10, 100 and 1000.

```
In [153]: def income_sample(df, sample_size):
              retDF = df.sample(n = sample_size, replace=True)
              mean_income = retDF['Income'].mean()
              print("Population income mean is", mean_income)
              plt.hist(retDF['Income'], bins=15, density=True, alpha=0.6, color='r', range=(0,210000))
              plt.xlabel("Income In USD")
              plt.ylabel("Probability Per Dollar")
              plt.title("Density Distribution Of Income")
              plt.scatter(mean_income, -.0000001, marker='^', color='red', s=300);
```

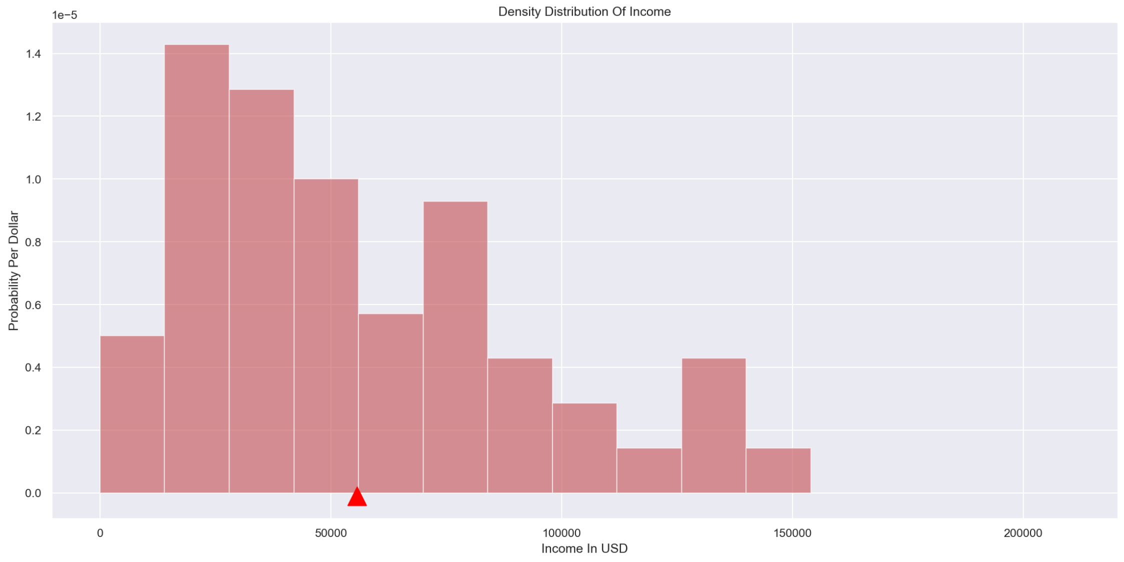
```
In [154]: income_sample(dfIncome,10)
```

```
Population income mean is 51571.1
```



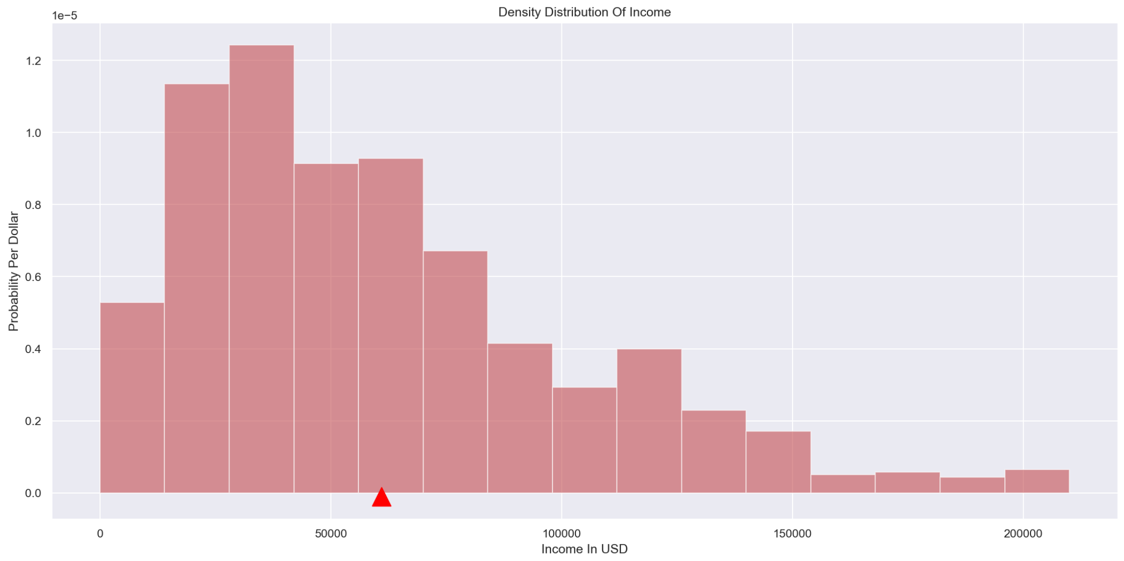
```
In [155]: income_sample(dfIncome,100)
```

```
Population income mean is 55664.23
```



```
In [156]: income_sample(dfIncome,1000)
```

```
Population income mean is 60995.746
```



Part 3cii). What happens to the shape of the empirical sample distributions of income as you increase the sample size?

As the sample size in this function grows, the modality of these distributions becomes **unimodal**. Also, when the sample size increases the distribution slowly tends to become more and more right skewed. The smallest sample size really doesn't have a distinguishable skew, but the larger ones have a skew.

QUESTION 3D:

If we want to estimate the **mean** of the population we can draw a sample from the population and compute the sample mean. As we learned in class, since samples can vary, the sample mean can vary and thus it is a random variable and has its own distribution.

i). Complete the function `income_sample_mean` below to randomly sample `sample_size` rows from `dfIncome` with replacement and return the sample mean of income for that sample.

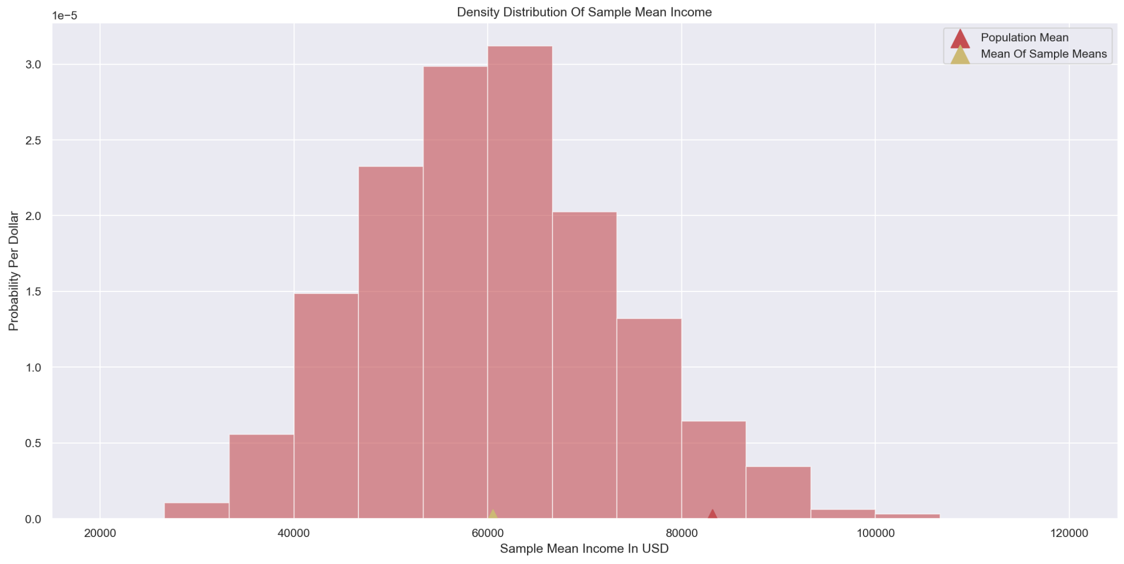
ii). Complete the function `income_sample_dist` below to simulate `num_simulations` of randomly sampling `sample_size` rows from `dfIncome` with replacement and calculate the sample mean of income for each sample. Store the sample means in an np.array called `means`. The function should output a **density** histogram of the empirical sample mean income distribution. On the histogram, include two markers on the histogram: A red one for the population mean (that you calculated in part 3A) and a yellow one for the mean of the `num_simulations` sample mean estimates. Include a title and labels for the x and y-axis.

Then run the cells provided below to output 3 separate distributions for `num_simulations`=1000 and `sample_size` = 10, 100 and 1000

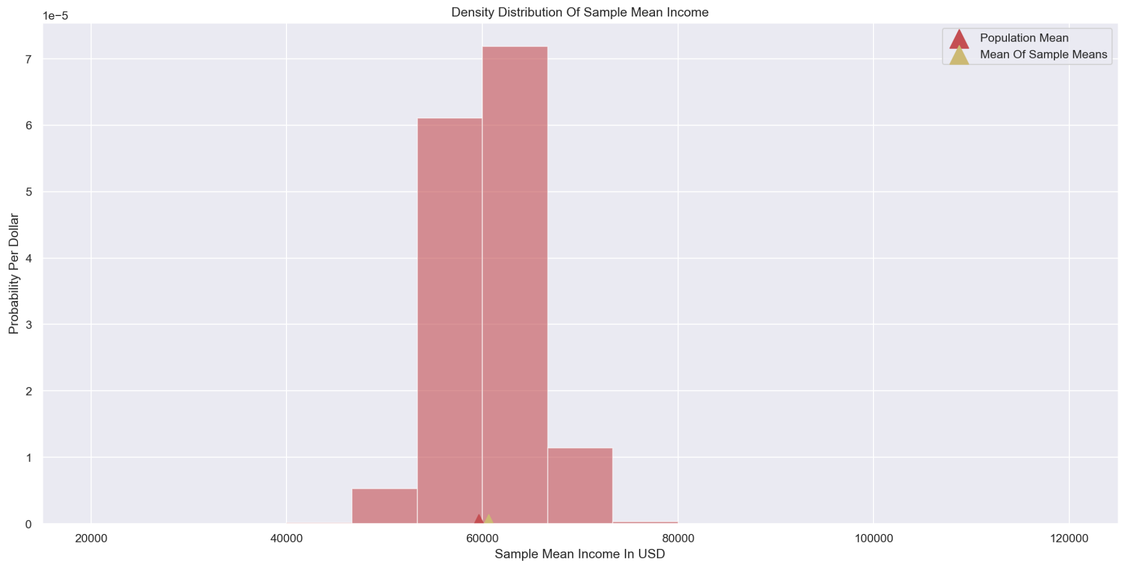
```
In [157]: def income_sample_mean(df, sample_size):
              retDF = df.sample(n = sample_size, replace=True)
              return retDF['Income'].mean()
```

```
In [158]: def income_sample_dist(df, sample_size, num_simulations):
              # 'means' stores "num_simulations" means from samples of size "sample_size"
              means = np.array([income_sample_mean(df, sample_size) for _ in range(num_simulations)])
              means_samples = np.mean(means)
              pop_mean = df.sample(n = sample_size, replace=True)['Income'].mean()
              plt.hist(means, bins=15, density=True, alpha=0.6, color='r', range=(20000,120000))
              plt.xlabel("Sample Mean Income In USD")
              plt.ylabel("Probability Per Dollar")
              plt.title("Density Distribution Of Sample Mean Income")
              plt.scatter(pop_mean, 0, marker='^', color='r', s=300, label='Population Mean');
              plt.scatter(means_samples, 0, marker='^', color='y', s=300, label='Mean Of Sample Means')
              plt.legend()

          # Your code for part (ii) above
```

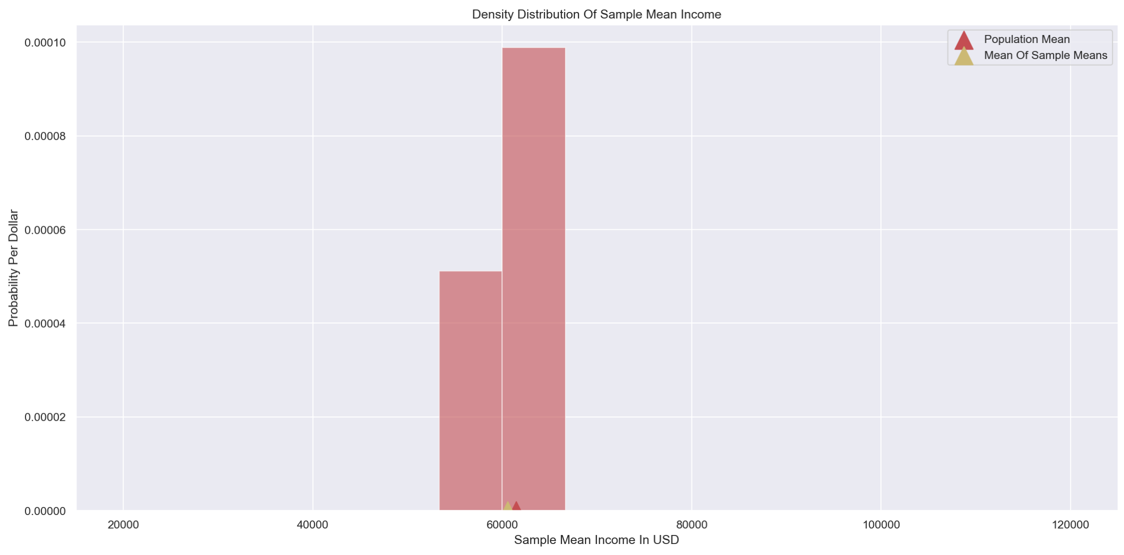
```
In [159]: income_sample_dist(dfIncome, 10, 1000)
```



```
In [160]: income_sample_dist(dfIncome, 100, 1000)
```



```
In [161]: income_sample_dist(dfIncome, 1000, 1000)
```


Density Distribution Of Sample Mean Income
Population Mean
Mean Of Sample Means
Probability Per Dollar
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
20000
40000
60000
80000
100000
120000
Sample Mean Income In USD

QUESTION 3E:

Describe the shapes of the empirical sample mean distributions (comment on their modality and skew compared to the modality and skew of the population distribution). What happens to the mean and standard deviations of these distributions as you increase the sample size? What is the name of the theorem that explains what you are observing?

As the sample size increases, the distribution becomes less and less normalized. With a small sample size, the distribution represents a normal distribution. With a larger sample size, the distribution becomes less normalized.

In a smaller sample size, the **modality** is unimodal and **slightly right skewed** (but not a lot). As the sample size increases, the **modality** is also unimodal but there is not a distinguishable **skew** in the distribution.

The mean of the population and the mean of the simulations become closer together in value when the sample size increases, the standard deviation also decreases. This is encapsulating the phenomena of the Central Limit Theorem.

QUESTION 4H.

Create an array called `simulated_statistics` that contains 50,000 simulated values of the test statistic under the null hypothesis. Assume that the original sample consisted of 210 experiments.

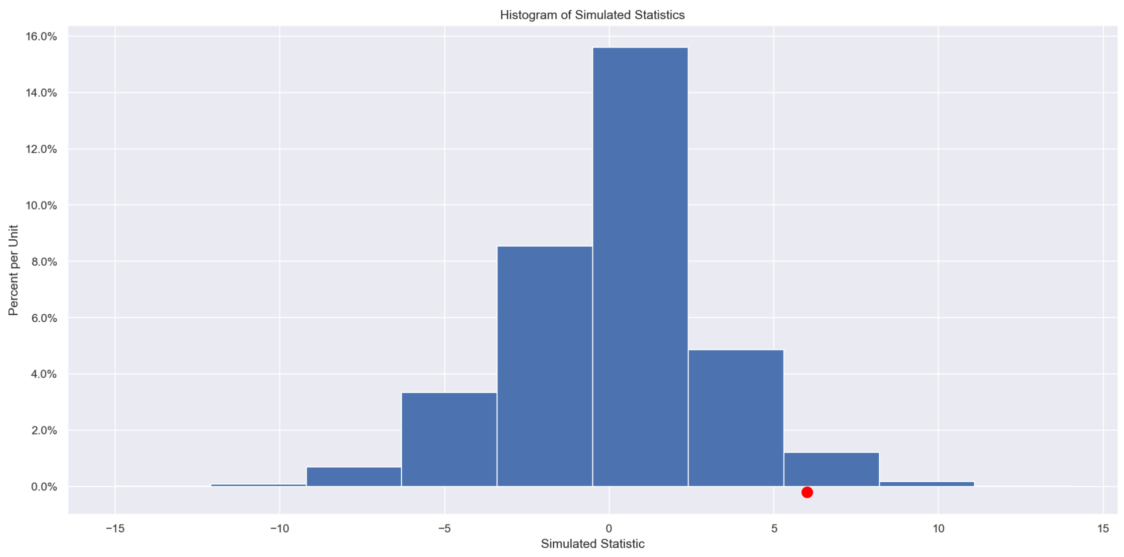
As usual, start by defining a function `one_simulated_statistic()` that simulates one value of the statistic. Your function should use `np.random.DISTRIBUTION` where `DISTRIBUTION` is the distribution you chose in part 4g. Your function should also use your `statistic` function from part 4e.

We have included the code that plots the distribution of the simulated values. The red dot represents the observed statistic you found in Question 4f.

```
In [176]: def one_simulated_statistic():
              n = 210
              p = 0.5
              num_correct = np.random.binomial(n, p)
              return int((num_correct / n) * 100 - 50)

          num_simulations = 50000

          simulated_statistics = np.array([one_simulated_statistic() for _ in range(num_simulations)])


          # Run the this cell a few times to see how the simulated statistic changes
          one_simulated_statistic()
```

```
Out[176]: -5
```

```
In [177]: # Run this cell to produce a histogram of the simulated statistics
          plt.hist(simulated_statistics, density = True, ec= "white")
          plt.xlabel('Simulated Statistic')
          plt.ylabel('Percent per Unit')
          plt.title('Histogram of Simulated Statistics')
          plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
          plt.scatter(observed_statistic, -0.002, color='red', s=100);
          plt.show()
```



Histogram of Simulated Statistics
Percent per Unit
16.0%
14.0%
12.0%
10.0%
8.0%
6.0%
4.0%
2.0%
0.0%
−15
−10
−5
0
5
10
15
Simulated Statistic