Final Exam - DSC 10, Fall 2022

Instructions:

- This exam consists of 12 questions, worth a total of 150 points.
- Write your PID in the top right corner of each page in the space provided.
- Please write **clearly** in the provided answer boxes; we will not grade work that appears elsewhere. Completely fill in bubbles and square boxes; if we cannot tell which option(s) you selected, you may lose points.

  ◯ A bubble means that you should only **select one choice**.

  ☐ A square box means you should **select all that apply**.
- You may refer to the Reference Sheet that was provided to you. Other than that, you may not refer to any other resources or technology during the exam (including calculators).

Full Name:

PID:

Lecture Section: ◯ A00 (10-10:50AM) ◯ B00 (11-11:50AM) ◯ C00 (1-1:50PM) ◯ D00 (12-12:50PM)

Seat you are in:

By signing below, you are agreeing that you will behave honestly and fairly during and after this exam.

Signature:

Version B

Please do not open your exam until instructed to do so.

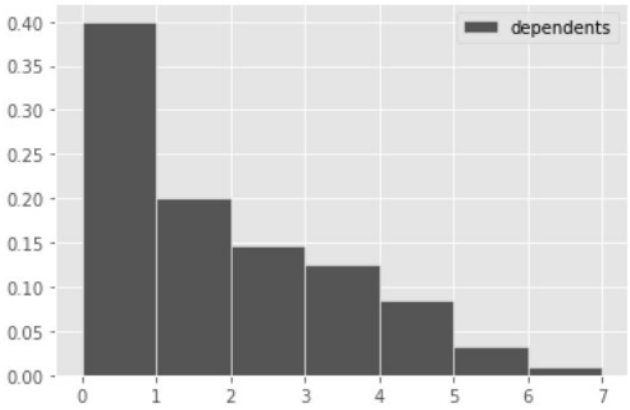
Make sure you have read the data description attached to the Reference Sheet!

Question 1 (26 pts)

In this question, we'll explore the number of dependents of each applicant. To begin, let's define the variable `dep_counts` as follows.

```
dep_counts = apps.groupby("dependents").count().get(["status"])
```

The visualization below shows the distribution of the number of dependents per applicant. Note that every applicant has 6 or fewer dependents.



Use `dep_counts` and the visualization above to answer the following questions.

a) (2 pts) What is the `type` of the variable `dep_counts`?

◯ array ◯ Series ◯ DataFrame

b) (2 pts) What type of data visualization is shown above?

◯ line plot ◯ scatter plot ◯ bar chart ◯ histogram

c) (4 pts) How many of the 1,000 applicants in `apps` have 2 or more dependents? Give your answer as an **integer**.

d) (4 pts) Define the DataFrame `dependents_status` as follows.

```
dependents_status = apps.groupby(["dependents", "status"]).count()
```

What is the maximum number of rows that `dependents_status` could have? Give your answer as an **integer**.

e) (5 pts) Recall that `dep_counts` is defined as follows.

```
dep_counts = apps.groupby("dependents").count().get(["status"])
```

Below, we define several more variables.

```
variable1 = dep_counts[dep_counts.get("status") >= 2].sum()

variable2 = dep_counts[dep_counts.index > 2].get("status").sum()

variable3 = (dep_counts.get("status").sum()
             - dep_counts[dep_counts.index < 2].get("status").sum())

variable4 = dep_counts.take(np.arange(2, 7)).get("status").sum()

variable5 = (dep_counts.get("status").sum()
             - dep_counts.get("status").loc[1]
             - dep_counts.get("status").loc[2])
```

Which of these variables are equal to your answer from part (c)? Select all that apply.

- ☐ `variable1`
- ☐ `variable2`
- ☐ `variable3`
- ☐ `variable4`
- ☐ `variable5`
- ☐ None of the above.

f) (5 pts) Next, we define variables `x` and `y` as follows.

```
x = dep_counts.index.values
y = dep_counts.get("status")
```

Note: If `idx` is the index of a Series or DataFrame, `idx.values` gives the values in `idx` **as an array**.

Which of the following expressions evaluate to the mean number of dependents? Select all that apply.

- ☐ `np.mean(x * y)`
- ☐ `x.sum() / y.sum()`
- ☐ `(x * y / y.sum()).sum()`
- ☐ `np.mean(x)`
- ☐ `(x * y).sum() / y.sum()`
- ☐ None of the above.

g) (4 pts) What does the expression `y.iloc[0] / y.sum()` evaluate to? Give your answer as a **fully simplified fraction**.



Question 2 (15 pts)

For each application in `apps`, we want to assign an age category based on the value in the `"age"` column, according to the table below.

`"age"`	age category
under 25	`"young adult"`
at least 25, but less than 50	`"middle aged"`
at least 50, but less than 75	`"older adult"`
75 or over	`"elderly"`

The incomplete code below should create a new DataFrame called `apps_cat` with all the data in `apps` plus a new column called `"age_category"` that contains the appropriate categorical description of each applicant's age.

```
cat_names = ["young adult", "middle aged", "older adult", "elderly"]

def age_to_bin(one_age):
    '''Returns the age category corresponding to one_age.'''
    one_age = __(a)__
    bin_pos = __(b)__
    return cat_names[bin_pos]

binned_ages = __(c)__
apps_cat = apps.assign(age_category = binned_ages)
```

a) (3 pts) Which of the following is a correct way to fill in blanks (a) and (b)?

	Blank (a)	Blank (b)
Option 1	`75 - one_age`	`round(one_age / 25)`
Option 2	`min(75, one_age)`	`one_age / 25`
Option 3	`75 - one_age`	`int(one_age / 25)`
Option 4	`min(75, one_age)`	`int(one_age / 25)`
Option 5	`min(74, one_age)`	`round(one_age / 25)`

◯ Option 1 ◯ Option 2 ◯ Option 3 ◯ Option 4 ◯ Option 5

b) (2 pts) Which of the following is a correct way to fill in blank (c)?

◯ `age_to_bin(apps.get("age"))`
◯ `apps.get("age").apply(age_to_bin)`
◯ `apps.get("age").age_to_bin()`
◯ `apps.get("age").apply(age_to_bin(one_age))`

c) (4 pts) Which of the following is a correct alternate implementation of the `age_to_bin` function? Select all that apply.

Option 1:

```
def age_to_bin(one_age):
    bin_pos = 3
    if one_age < 25:
        bin_pos = 0
    if one_age < 50:
        bin_pos = 1
    if one_age < 75:
        bin_pos = 2
    return cat_names[bin_pos]
```

Option 2:

```
def age_to_bin(one_age):
    bin_pos = 3
    if one_age < 75:
        bin_pos = 2
    if one_age < 50:
        bin_pos = 1
    if one_age < 25:
        bin_pos = 0
    return cat_names[bin_pos]
```

Option 3:

```
def age_to_bin(one_age):
    bin_pos = 0
    for cutoff in np.arange(25, 100, 25):
        if one_age >= cutoff:
            bin_pos = bin_pos + 1
    return cat_names[bin_pos]
```

Option 4:

```
def age_to_bin(one_age):
    bin_pos = -1
    for cutoff in np.arange(0, 100, 25):
        if one_age >= cutoff:
            bin_pos = bin_pos + 1
        return cat_names[bin_pos]
```

☐ Option 1 ☐ Option 2 ☐ Option 3 ☐ Option 4 ☐ None of the above.

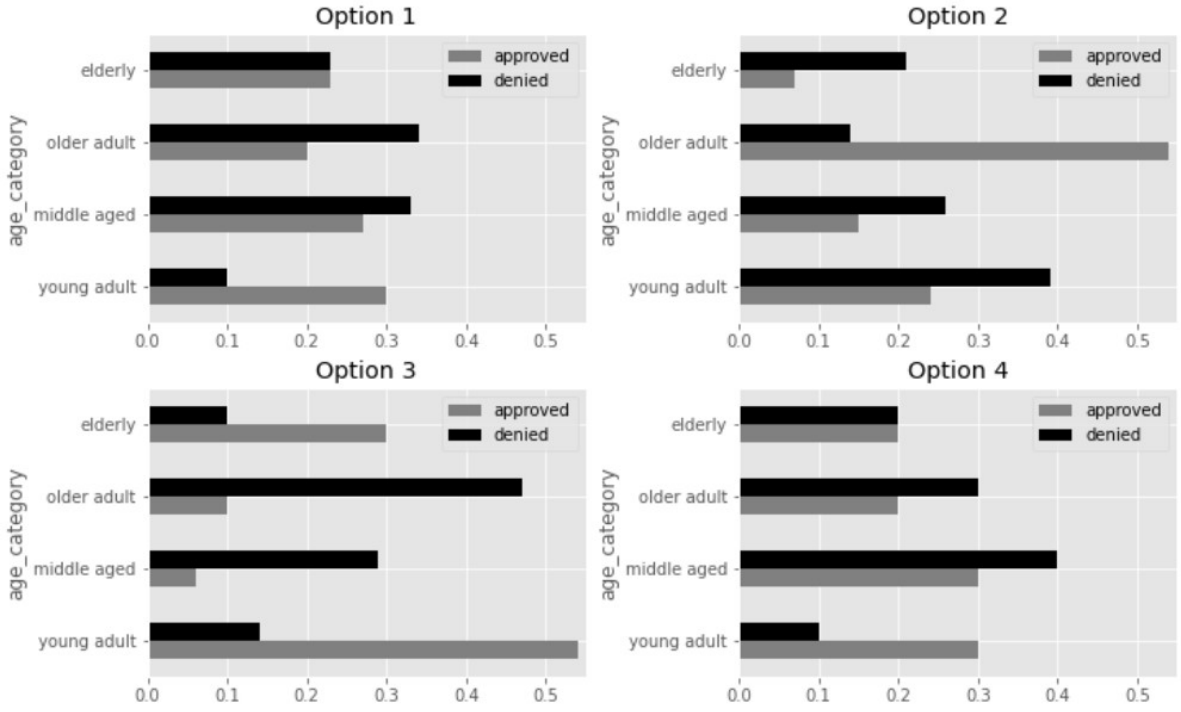
d) (3 pts) We want to determine the number of `"middle aged"` applicants whose applications were denied. Fill in the blank below so that `count` evaluates to that number.

```
df = apps_cat._________.reset_index()
count = df[(df.get("age_category") == "middle aged") &
           (df.get("status") == "denied")].get("income").iloc[0]
```

What goes in the blank?

e) (3 pts) The total variation distance between the distributions of `"age_category"` for approved applications and denied applications is 0.4.

One of the visualizations below shows the distributions of `"age_category"` for approved applications and denied applications. Which visualization is it?



◯ Option 1 ◯ Option 2 ◯ Option 3 ◯ Option 4

PID: _____

Question 3 (12 pts)

In `apps`, our sample of 1,000 credit card applications, 500 of the applications come from homeowners and 500 come from people who don't own their own home. In this sample, homeowner ages have a mean of 40 and standard deviation of 10. We want to use the bootstrap method to compute a confidence interval for the mean age of a homeowner in the population of all credit card applicants.

a) (4 pts) Suppose our computer is too slow to bootstrap 10,000 times, and instead can only bootstrap 20 times. Here are the 20 resample means, sorted in ascending order:

$$37, 38, 39, 39, 40, 40, 40, 40, 41, 41,$$
$$42, 42, 42, 42, 42, 42, 43, 43, 43, 44$$

What are the left and right endpoints of a bootstrapped **80%** confidence interval for the population mean? Use the mathematical definition of percentile.

Left endpoint = ☐ Right endpoint = ☐

b) (2 pts) True or False: Using the mathematical definition of percentile, the 50th percentile of the bootstrapped distribution above equals its median.

◯ True ◯ False

c) (4 pts) Consider the following three quantities:

1. `pop_mean`, the unknown mean age of homeowners in the population of all credit card applicants.
2. `sample_mean`, the mean age of homeowners in our sample of 500 applications in `apps`. We know this is 40.
3. `resample_mean`, the mean age of homeowners in one particular resample of the applications in `apps`.

Which of the following statements about the relationship between these three quantities are guaranteed to be true? Select all that apply.

☐ If `sample_mean` is less than `pop_mean`, then `resample_mean` is also less than `pop_mean`.
☐ The mean of `sample_mean` and `resample_mean` is closer to `pop_mean` than either of the two values individually.
☐ `resample_mean` is closer than `sample_mean` to `pop_mean`.
☐ `resample_mean` is further than `sample_mean` from `pop_mean`.
☐ None of the above.

d) (2 pts) Which of the following is approximately equal to the standard deviation of homeowner ages in the population?

◯ $\frac{10}{\sqrt{500}}$ ◯ 10 ◯ 40 ◯ $10 \cdot \sqrt{500}$

Question 4 (15 pts)

In `apps`, our sample of 1,000 credit card applications, applicants who were approved for the credit card have fewer dependents, on average, than applicants who were denied. The mean number of dependents for approved applicants is 0.98, versus 1.07 for denied applicants.

To test whether this difference is purely due to random chance, or whether the distributions of the number of dependents for approved and denied applicants are truly different in the population of all credit card applications, we decide to perform a permutation test.

Consider the incomplete code block below.

```
def shuffle_status(df):
    shuffled_status = np.random.permutation(df.get("status"))
    return df.assign(status=shuffled_status).get(["status", "dependents"])

def test_stat(df):
    grouped = (df.get(["status", "dependents"])
                 .groupby("status").mean().get("dependents"))
    approved = grouped.loc["approved"]
    denied = grouped.loc["denied"]
    return __(a)__

stats = np.array([])
for i in np.arange(10000):
    shuffled_apps = shuffle_status(apps)
    stat = test_stat(shuffled_apps)
    stats = np.append(stats, stat)

p_value = np.count_nonzero(__(b)__) / 10000
```

Below are six options for filling in blanks (a) and (b) in the code above.

	Blank (a)	Blank (b)
Option 1	`denied - approved`	`stats >= test_stat(apps)`
Option 2	`denied - approved`	`stats <= test_stat(apps)`
Option 3	`approved - denied`	`stats >= test_stat(apps)`
Option 4	`np.abs(denied - approved)`	`stats >= test_stat(apps)`
Option 5	`np.abs(denied - approved)`	`stats <= test_stat(apps)`
Option 6	`np.abs(approved - denied)`	`stats >= test_stat(apps)`

The correct way to fill in the blanks depends on how we choose our null and alternative hypotheses.

a) (3 pts) Suppose we choose the following pair of hypotheses.

- **Null Hypothesis**: In the population, the number of dependents of accepted and denied applicants come from the same distribution.
- **Alternative Hypothesis**: In the population, the number of dependents of accepted applicants and denied applicants do not come from the same distribution.

Which of the six presented options could correctly fill in blanks (a) and (b) for this pair of hypotheses? Select all that apply.

☐ Option 1
☐ Option 2
☐ Option 3
☐ Option 4
☐ Option 5
☐ Option 6
☐ None of the above.

b) (3 pts) Now, suppose we choose the following pair of hypotheses.

- **Null Hypothesis**: In the population, the number of dependents of accepted and denied applicants come from the same distribution.
- **Alternative Hypothesis**: In the population, the number of dependents of accepted applicants is smaller on average than the number of dependents of denied applicants.

Which of the six presented options could correctly fill in blanks (a) and (b) for this pair of hypotheses? Select all that apply.

☐ Option 1
☐ Option 2
☐ Option 3
☐ Option 4
☐ Option 5
☐ Option 6
☐ None of the above.

c) (3 pts) Option 6 from the start of this question is repeated below.

	Blank (a)	Blank (b)
Option 6	`np.abs(approved - denied)`	`stats >= test_stat(apps)`

We want to create a new option, Option 7, that replicates the behavior of Option 6, but with blank (a) filled in as shown:

	Blank (a)	Blank (b)
Option 7	`approved - denied`	

Which expression below could go in blank (b) so that Option 7 is equivalent to Option 6?

◯ `np.abs(stats) >= test_stat(apps)`

◯ `stats >= np.abs(test_stat(apps))`

◯ `np.abs(stats) >= np.abs(test_stat(apps))`

◯ `np.abs(stats >= test_stat(apps))`

d) (6 pts) In our implementation of this permutation test, we followed the procedure outlined in lecture to draw new pairs of samples under the null hypothesis and compute test statistics — that is, we randomly assigned each row to a group (approved or denied) by shuffling one of the columns in `apps`, then computed the test statistic on this random pair of samples.

Let's now explore an alternative solution to drawing pairs of samples under the null hypothesis and computing test statistics. Here's the approach:

1. Shuffle, i.e. re-order, the rows of the DataFrame.
2. Use the values at the top of the resulting `"dependents"` column as the new "denied" sample, and the values at the at the bottom of the resulting `"dependents"` column as the new "approved" sample. Note that we don't necessarily split the DataFrame exactly in half — the sizes of these new samples depend on the number of "denied" and "approved" values in the original DataFrame!

Once we generate our pair of random samples in this way, we'll compute the test statistic on the random pair, as usual. Here, we'll use as our test statistic the difference between the mean number of dependents for denied and approved applicants, in the order **denied minus approved**.

Fill in the blanks to complete the simulation below.

Hint: `np.random.permutation` shouldn't appear anywhere in your code.

```
def shuffle_all(df):
    '''Returns a DataFrame with the same rows as df, but reordered.'''
    return __(a)__

def fast_stat(df):
    # This function does not and should not contain any randomness.
    denied = np.count_nonzero(df.get("status") == "denied")
    mean_denied = __(b)__.get("dependents").mean()
    mean_approved = __(c)__.get("dependents").mean()
    return mean_denied - mean_approved

stats = np.array([])
for i in np.arange(10000):
    stat = fast_stat(shuffle_all(apps))
    stats = np.append(stats, stat)
```

What goes in blank (a)?

What goes in blank (b)?

What goes in blank (c)?

Question 5 (10 pts)

Choose the best tool to answer each of the following questions. Note the following:

- By "hypothesis testing," we mean "standard" hypothesis testing, i.e. hypothesis testing that **doesn't** involve permutation testing or bootstrapping.
- By "bootstrapping," we mean bootstrapping that **doesn't** involve hypothesis testing.

a) (2 pts) Are incomes of applicants with 2 or fewer dependents drawn randomly from the distribution of incomes of all applicants?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

b) (2 pts) What is the median income of credit card applicants with 2 or fewer dependents?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

c) (2 pts) Are credit card applications approved through a random process in which 50% of applications are approved?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

d) (2 pts) Is the median income of applicants with 2 or fewer dependents less than the median income of applicants with 3 or more dependents?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

e) (2 pts) What is the difference in median income of applicants with 2 or fewer dependents and applicants with 3 or more dependents?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

PID: _____

Question 6 (16 pts)

In this question, we'll explore the relationship between the ages and incomes of credit card applicants.

a) (4 pts) The credit card company that owns the data in `apps`, BruinCard, has decided not to give us access to the entire `apps` DataFrame, but instead just a sample of `apps` called `small_apps`.

We'll start by using the information in `small_apps` to compute the regression line that predicts the age of an applicant given their income.

For an applicant with an income that is $\frac{8}{3}$ standard deviations above the mean income, we predict their age to be $\frac{4}{5}$ standard deviations above the mean age. What is the correlation coefficient, r, between incomes and ages in `small_apps`? Give your answer as a **fully simplified fraction**.

$r =$ ☐

b) (4 pts) Now, we want to predict the income of an applicant given their age. We will again use the information in `small_apps` to find the regression line.

The regression line predicts that an applicant whose age is $\frac{4}{5}$ standard deviations above the mean age has an income that is s standard deviations above the mean income.

What is the value of s? Give your answer as a **fully simplified fraction**.

$s =$ ☐

c) (4 pts) BruinCard has now taken away our access to both `apps` and `small_apps`, and has instead given us access to an even smaller sample of `apps` called `mini_apps`. In `mini_apps`, we know the following information:

- All incomes and ages are positive numbers.
- There is a positive linear association between incomes and ages.

We use the data in `mini_apps` to find the regression line that will allow us to predict the income of an applicant given their age. Just to test the limits of this regression line, we use it to predict the income of an applicant who is **-2 years old**, even though it doesn't make sense for a person to have a negative age.

Let I be the regression line's prediction of this applicant's income. Which of the following inequalities are guaranteed to be satisfied? Select all that apply.

☐ $I < 0$

☐ $I <$ mean income

☐ $|I - \text{mean income}| \leq |\text{mean age} + 2|$

☐ $\dfrac{|I - \text{mean income}|}{\text{standard deviation of incomes}} \leq \dfrac{|\text{mean age} + 2|}{\text{standard deviation of ages}}$

☐ None of the above.

d) (4 pts) Yet again, BruinCard, the company that gave us access to `apps`, `small_apps`, and `mini_apps`, has revoked our access to those three DataFrames and instead has given us `micro_apps`, an even smaller sample of `apps`.

Using `micro_apps`, we are again interested in finding the regression line that will allow us to predict the income of an applicant given their age. We are given the following information:

- The correlation coefficient, r, between ages and incomes is $-\frac{1}{3}$ (note the negative sign).
- The mean income is $\frac{7}{2}$ (remember, incomes are measured in tens of thousands of dollars).
- The mean age is 33.
- The regression line predicts that a 24 year old applicant has an income of $\frac{31}{2}$.

Suppose the standard deviation of incomes in `micro_apps` is an integer multiple of the standard deviation of ages in `micro_apps`. That is,

$$\text{standard deviation of income} = k \cdot \text{standard deviation of age.}$$

What is the value of k? Give your answer as an **integer**.

$k =$ ☐

PID: _____

Question 7 (12 pts)

Below, we define a new DataFrame called `seven_apps` and display it fully.

```
seven_apps = apps.sample(7).sort_values(by="dependents",
                                        ascending=False)
seven_apps
```

	status	age	income	homeowner	dependents
505	approved	52.16667	2.6600	yes	3
474	approved	39.00000	3.5000	yes	2
934	approved	22.25000	2.8000	yes	1
828	approved	21.41667	1.5896	no	0
970	approved	21.83333	2.0272	no	0
18	denied	35.58333	4.0000	no	0
784	approved	32.83333	2.5000	no	0

Consider the process of **resampling 7 rows from `seven_apps` with replacement**, and computing the maximum number of dependents in the resample.

a) (4 pts) If we take one resample, what is the probability that the maximum number of dependents in the resample is **less than** 3? Leave your answer **unsimplified**.

b) (4 pts) If we take 50 resamples, what is the probability that the maximum number of dependents is **never** 3, in any resample? Leave your answer **unsimplified**.

c) (4 pts) If we take 50 resamples, what is the probability that the maximum number of dependents is 3 in **every** resample? Leave your answer **unsimplified**.

Question 8 (12 pts)

At TritonCard, a new UCSD alumni-run credit card company, applications are approved at random. Each time someone submits an application, a TritonCard employee rolls a fair six-sided die two times. If both rolls are the same even number — that is, if both are 2, both are 4, or both are 6 — TritonCard approves the application. Otherwise, they reject it.

a) (4 pts) You submit k identical TritonCard applications. The probability that at least one of your applications is approved is of the form

$$1 - \left(\frac{a}{b}\right)^k.$$

What are the values of a and b? Give your answers as **integers** such that the fraction $\frac{a}{b}$ is **fully simplified**.

$a =$ $\qquad$ $b =$

b) (4 pts) Every TritonCard credit card has a 3-digit security code on the back, where each digit is a number 0 through 9. There are 1,000 possible 3-digit security codes: $000, 001, \ldots, 998, 999$.

Tony, the CEO of TritonCard, wants to only issue credit cards whose security codes satisfy all of the following criteria:

1. The first digit is odd.
2. The middle digit is 0.
3. The last digit is even.

Tony doesn't have a great way to generate security codes meeting these three criteria, but he does know how to generate security codes with **three *unique* (distinct) digits**. That is, no number appears in the security code more than once. So, Tony decides to randomly select a security code from among those with three unique digits. If the randomly selected security code happens to meet the desired criteria, TritonCard will issue a credit card with this security code, otherwise Tony will try his process again.

What is the probability that Tony's first randomly selected security code satisfies the given criteria? Give your answer as a **fully simplified fraction**.

c) (4 pts) Daphne, the newest employee at TritonCard, wants to try a different way of generating security codes. She decides to randomly select a 3-digit code from among **all 1,000** possible security codes (i.e. the digits are not necessarily unique). As before, if the code randomly selected code happens to meet the desired criteria, TritonCard will issue a credit card with this security code, otherwise Daphne will try her process again.

What is the probability that Daphne's first randomly selected security code satisfies the given criteria? Give your answer as a **fully simplified fraction**.



Question 9 (6 pts)

After you graduate, you are hired by TritonCard! On your new work computer, you install `numpy`, but something goes wrong with the installation — your copy of `numpy` doesn't come with `np.random.multinomial`. To demonstrate your resourcefulness to your new employer, you decide to implement your own version of `np.random.multinomial`.

Below, complete the implementation of the function `manual_multinomial` so that `manual_multinomial(n, p)` works the same way as `np.random.multinomial(n, p)`. That is, `manual_multinomial` should take in an integer `n` and an array of probabilities `p`. It should return an array containing the counts in each category when we randomly draw `n` items from a categorical distribution where the probabilities of drawing an item from each category are given in the array `p`. The array returned by `manual_multinomial(n, p)` should have a length of `len(p)` and a sum of `n`.

For instance, to simulate flipping a coin five times, we could call `manual_multinomial(5, np.array([0.5, 0.5]))`, and the output might look like `array([2, 3])`.

```
def manual_multinomial(n, p):
    values = np.arange(len(p))
    choices = np.random.choice(values, size=__(a)__, replace=__(b)__, p=p)
    value_counts = np.array([])
    for value in values:
        value_count = __(c)__
        value_counts = np.append(value_counts, value_count)
    return value_counts
```

What goes in blank (a)?

What goes in blank (b)?

What goes in blank (c)?

PID: _____

Question 10 (11 pts)

The credit card company that owns the data in `apps`, BruinCard, has decided not to give us access to the entire `apps` DataFrame, but instead just a random sample of 100 rows of `apps` called `hundred_apps`.

We are interested in estimating the mean age of all applicants in `apps` given only the data in `hundred_apps`. The ages in `hundred_apps` have a mean of 35 and a standard deviation of 10.

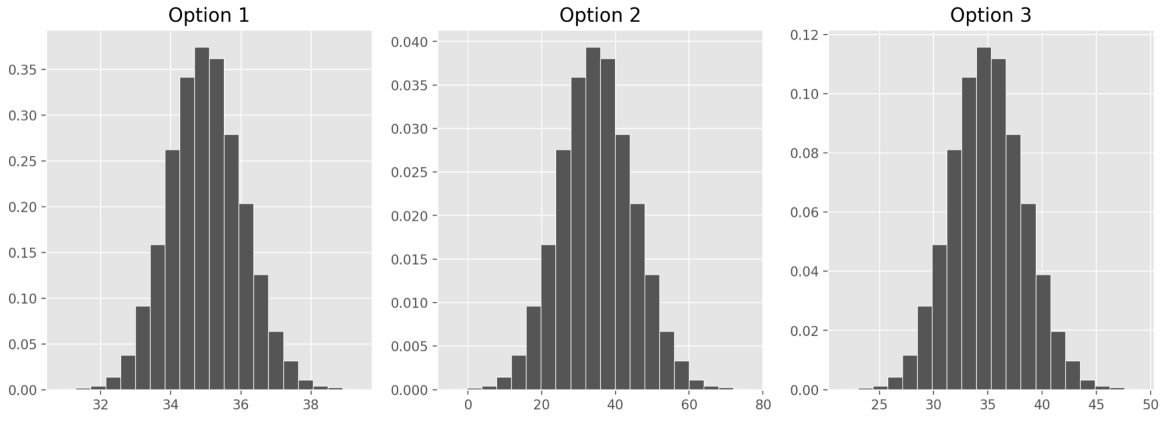
a) (4 pts) Give the endpoints of the CLT-based 95% confidence interval for the mean age of all applicants in `apps`, based on the data in `hundred_apps`.

Left endpoint = ____ Right endpoint = ____

BruinCard reinstates our access to `apps` so that we can now easily extract information about the ages of all applicants. We determine that, just like in `hundred_apps`, the ages in `apps` have a mean of 35 and a standard deviation of 10. This raises the question of how other samples of 100 rows of `apps` would have turned out, so we compute 10,000 sample means as follows.

```
sample_means = np.array([])
for i in np.arange(10000):
    sample_mean = apps.sample(100, replace=True).get("age").mean()
    sample_means = np.append(sample_means, sample_mean)
```

b) (2 pts) Which of the following three visualizations best depict the distribution of `sample_means`?



◯ Option 1 ◯ Option 2 ◯ Option 3

c) (5 pts) Which of the following statements are guaranteed to be true? Select all that apply.

- ☐ We used bootstrapping to compute `sample_means`.
- ☐ The ages of credit card applicants are roughly normally distributed.
- ☐ A CLT-based 90% confidence interval for the mean age of credit card applicants, based on the data in `hundred_apps`, would be narrower than the interval you gave in part (a).
- ☐ The expression `np.percentile(sample_means, 2.5)` evaluates to the left endpoint of the interval you gave in part (a).
- ☐ If we used the data in `hundred_apps` to create 1,000 CLT-based 95% confidence intervals for the mean age of applicants in `apps`, approximately 950 of them would contain the true mean age of applicants in `apps`.
- ☐ None of the above.

PID: _____

Question 11 (12 pts)

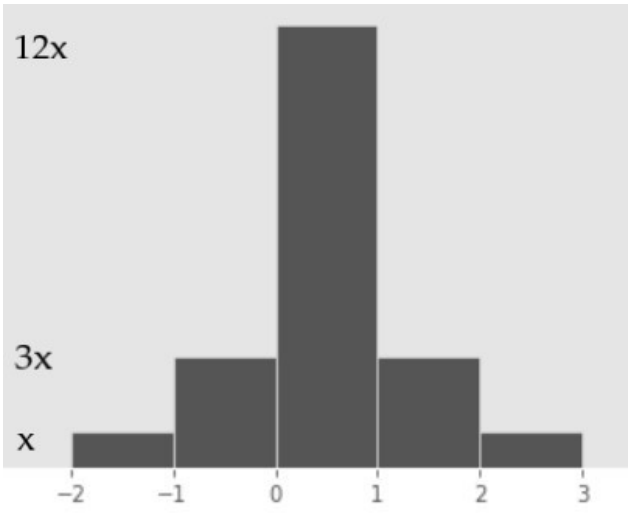
Suppose variables `v1`, `v2`, `v3`, and `v4`, have already been initialized to specific numerical values. Right now, we don't know what values they've been set to.

The function `f` shown below takes in a number, `v`, and outputs an integer between -2 and 2, depending on the value of `v` relative to `v1`, `v2`, `v3`, and `v4`.

```
def f(v):
    if v <= v1:
        return -2
    elif v <= v2:
        return -1
    elif v <= v3:
        return 0
    elif v <= v4:
        return 1
    else:
        return 2
```

Recall that in the previous question, we created an array called `sample_means` containing 10,000 values, each of which is the mean of a random sample of 100 applicant ages drawn from the DataFrame `apps`, in which ages have a mean of 35 and a standard deviation of 10.

When we call the function `f` on every value `v` in `sample_means`, we produce a collection of 10,000 values all between -2 and 2. A density histogram of these values is shown below.



The heights of the five bars in this histogram, reading from left to right, are

$$x,\ 3x,\ 12x,\ 3x,\ x.$$

a) (4 pts) What is the value of x (i.e. the height of the shortest bar in the histogram)? Give your answer as a **fully simplified fraction**.

$x =$ ____

b) (4 pts) What does the expression below evaluate to? Give your answer as an **integer**.

```
np.count_nonzero((sample_means > v2) & (sample_means <= v4))
```

Hint: Don't try to find the values of `v2` and `v4` – you can answer this question without them!

____

Suppose we have run the code below.

```
from scipy import stats

def g(u):
    return stats.norm.cdf(u) - stats.norm.cdf(-u)
```

Several input-output pairs for the function `g` are shown in the table below. Some of them will be useful to you in answering the questions that follow.

u	g(u)
0.84	0.60
1.28	0.80
1.65	0.90
2.25	0.975

c) (2 pts) What is the value of `v3`, one of the variables used in the function `f`? Give your answer as a number.

Hint: Use the histogram as well as one of the rows of the table above.

`v3` = ____

d) (2 pts) Which of the following is closest to the value of the expression below?

```
np.percentile(sample_means, 5)
```

Hint: Use the histogram as well as one of the rows of the table above.

◯ 14 ◯ 14.75 ◯ 33 ◯ 33.35 ◯ 33.72

Question 12 (3 pts)

Feel free to draw us a picture or tell us about your fondest memory from DSC 10 this quarter.

Before turning in your exam, please make sure that your PID is on every page.

Congratulations on finishing DSC 10!

The Data

Credit cards allow consumers to make purchases by borrowing money and paying it back later. Credit card companies are wary of granting this borrowing ability to consumers who may not be able to pay back their debt. Therefore, potential credit card carriers must submit an application that contains information about themselves and their history of paying back debt.

The DataFrame `apps` contains application data for a random sample of 1,000 applicants for a particular credit card from the 1990s. The columns are:

- `"status" (str)`: Whether the credit card application was approved: `"approved"` or `"denied"` values only.
- `"age" (float)`: The applicant's age, in years, to the nearest twelfth of a year.
- `"income" (float)`: The applicant's annual income, in tens of thousands of dollars.
- `"homeowner" (str)`: Whether the credit card applicant owns their own home: `"yes"` or `"no"` values only.
- `"dependents" (int)`: The number of dependents, or individuals that rely on the applicant as a primary source of income, such as children.

The first few rows of `apps` are shown below, though remember that `apps` has 1,000 rows.

	status	age	income	homeowner	dependents
0	denied	23.25000	1.6200	no	0
1	approved	30.41667	4.1500	no	1
2	approved	39.41667	4.7000	yes	4
3	approved	44.08333	4.0000	yes	2
4	approved	51.75000	4.7232	yes	1
5	approved	32.00000	2.4000	no	1

Throughout this exam, we will refer to `apps` repeatedly.

Assume that:

- Each applicant only submitted a single application.
- We have already run `import babypandas as bpd` and `import numpy as np`.