15-110 Check6-2 - Written Portion

Name:

AndrewID:

#1 - Recognize Analyses - 6pts

Each of the following snippets of code performs a basic data analysis that we discussed in class. Select what each snippet does from the options provided below.

```
# assume scores is a 1D list of integers
scores.sort()
print(scores[len(scores) // 2])
```

☐ mean	☐ bucketing	☐ removes duplicates
☐ median	☐ basic probability	☐ marks missing data
☐ mode	☐ joint probability	☐ removes outliers

```
# assume petList is a list of dictionaries, data entries about pets
adoptedDogCount = 0
for pet in petList:
    if pet["species"] == "dog" and pet["status"] == "adopted":
        adoptedDogCount += 1
print(adoptedDogCount / len(PetList))
```

☐ mean	☐ bucketing	☐ removes duplicates
☐ median	☐ basic probability	☐ marks missing data
☐ mode	☐ joint probability	☐ removes outliers

```
# assume data is a 2D list of data entries
i = 0
while i < len(data):
    if data[i] in data[:i]:
        data.pop(i)
    else:
        i += 1
```

☐ mean	☐ bucketing	☐ removes duplicates
☐ median	☐ basic probability	☐ marks missing data
☐ mode	☐ joint probability	☐ removes outliers

#2 - Matplotlib - 9pts

For each of the following lines of matplotlib code from lecture, write a short statement that explains **at a high level** what the code does. Some lines of code have been modified slightly.

```
ax.hist(data, bins=5)
```

```
flavors = [ "vanilla", "chocolate", "strawberry" ]
ax.set_xticklabels(flavors)
```

```
menMeans = [20, 35, 30, 35, 27]
menStd = [2, 3, 4, 1, 2]
mensInd = np.arange(5)
rects1 = ax.bar(mensInd, menMeans, 20, yerr=menStd)
```

#3 - Monte Carlo Methods - 9pts

For each of the following questions, use **Monte Carlo methods** to find the answer to the given question. You can use the `monteCarlo(trials)` function from the notes to average results over `100,000` trials; you just need to update the `runTrial()` function for each question.

Please submit your answer as a decimal probability (like 0.45; 100% = 1, 50% = 0.5), and round your answer for each question to have only 2 digits after the decimal point.

What is the probability that, if you roll a die twice, the second roll will be either 2 larger or 2 smaller than the first?

For example, you could roll a 4 and then a 6, or a 4 and then a 2.

Pick a random odd number between 1 and 99. What is the probability that that number is a multiple of 7?

Hint: make a list of all odd numbers between 1 and 99, then use `random.choice()`

Make a list with six values (two `"red"`, two `"green"`, two `"blue"`) and shuffle it. What is the probability that the first two values in the list are both `"red"`?

Hint: use the destructive function `random.shuffle()`

#4 - Advanced Simulation - 6pts

Recall the zombie outbreak simulation we wrote in class. The following questions test your understanding of how all the code works together.

Which of the following segments of code set up the original number of zombies?

- ☐ `for zombie in range(5):  data["creatures"].append(...`
- ☐ `move = random.choice([[-1, 0], [1, 0], [0, -1], [0, 1]])`
- ☐ `creature["species"] = "zombie"`

Which of the following segments of code made a zombie move in a random direction?

- ☐ `row = creature["row"] ; col = creature["col"]`
- ☐ `creature["row"] += move[0] ; creature["col"] += move[1]`
- ☐ `zombiePositions.append([creature["row"], creature["col"]])`

Which of the following segments of code determined whether or not a specific human was infected?

- ☐ `data["rate"] = 0.5`
- ☐ `if creature["species"] == "human":    color = "purple" ;`
  `else:    color = "green"`
- ☐ `odds = random.random() ; if odds < data["rate"]:  ...`

#5 - Game Trees - 9pts

Nim (https://en.wikipedia.org/wiki/Nim) is a simple game for two players. The game starts with a pot containing some number of marbles on the table. Players take turns removing marbles from the pot. Each player must choose to remove 1, 2, or 3 marbles on their turn. Whoever removes the last marble loses.

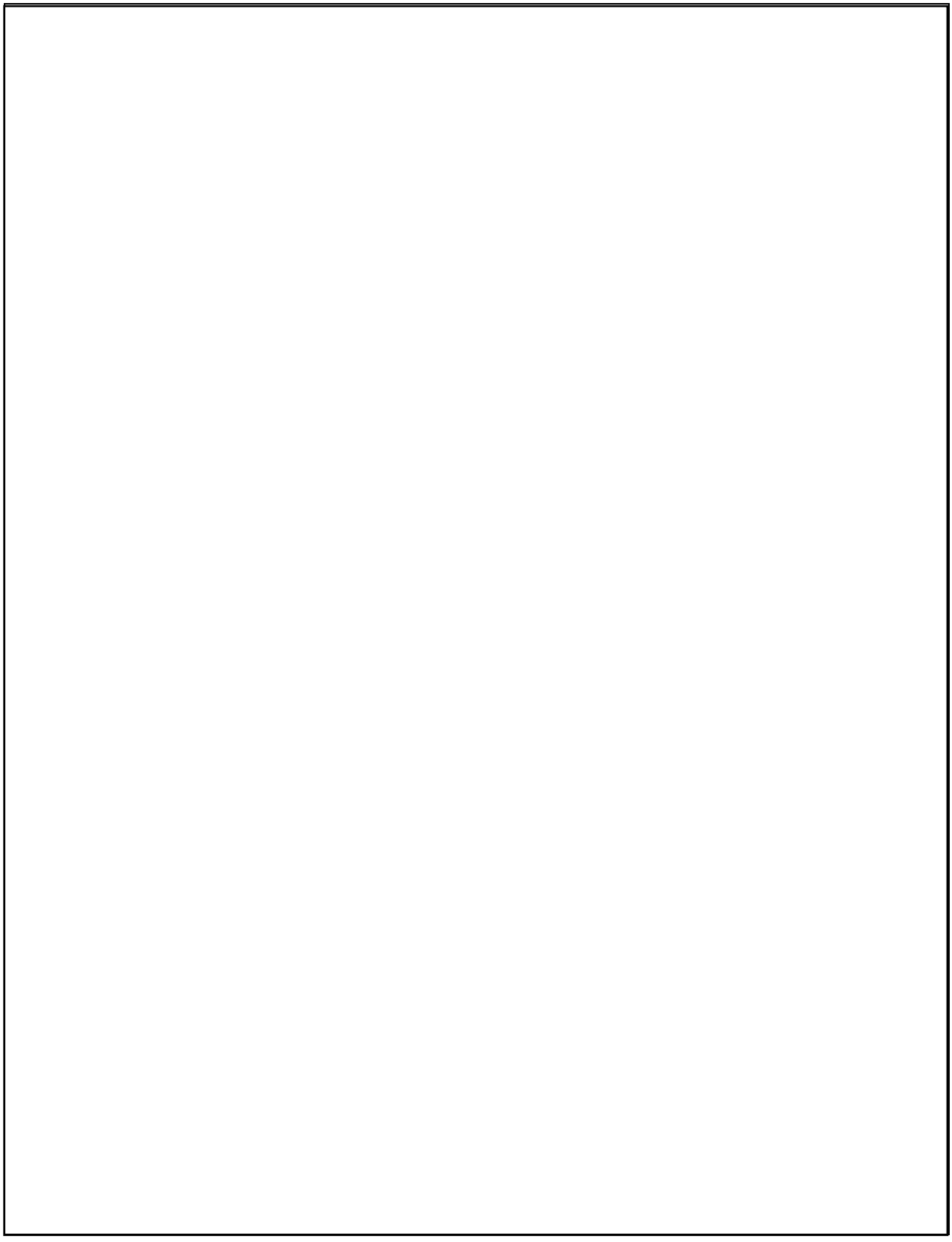
Assume you want to build a basic AI that can play Nim using a game tree. In this game, the pot will start with 16 marbles, and the **state** of the game is the number of marbles in the pot. On the next page, draw the root node and the first two levels of the game tree (you do not have to draw any levels past that), with the number of marbles as the value of each node. Annotate your game tree to show which actions are taken by the AI vs. the opponent, assuming the AI gets the first turn.

You can do this with a picture of a physical drawing or an online image editing tool (like Google Drawings). To upload the image, use the same approach you used on Hw5.

What is the maximum **depth** of this game tree? (A node with only the root has depth 1).

Assume that the AI uses minimax to find the best action to take on its turn. When comparing the results on the level right below the root, should the AI pass the **maximum** or **minimum** result to the top level with the root?

- ☐ Maximum
- ☐ Minimum



#6 - Heuristics - 6pts

Consider the two-player game **Draughts**, or **Checkers** (https://en.wikipedia.org/wiki/Draughts). This game is too complex for an AI to build a full game tree; it would need to use **heuristics** instead, to support fast search for the next move to make.

Which of the following factors could be included to build a well-designed heuristic for a Draughts game state? **Select all that apply.**

- ❑ How many pieces each player has on the board
- ❑ How many 'king' pieces each player has on the board
- ❑ Whether the players are calling the game 'draughts' or 'checkers'
- ❑ Where pieces are located (closer to the opponent's side = better)

Which of the following **best** describes how an AI could choose its next move, using that heuristic? **Select only one answer.**

- ❑ Apply the heuristic to the current board, and use the resulting score to choose which move to take
- ❑ Build a game tree to some set depth, score the leaves with the heuristic algorithm, and then apply minimax to get the result
- ❑ Build the next level of the game tree, then apply the heuristic to find the best-scoring child. Then generate all of that child's next moves, and apply the heuristic again. Continue until an end state is reached to get the next move
- ❑ Choose one of the possible moves randomly