AP Computer Science A Extra Problems

As the semester winds down and you work on your projects, there are several other opportunities for you to earn extra scores in the gradebook. Below are several problems with associated point values. Solving any of the problems will earn you an extra perfect score in the Gradebook (not quite sure yet how many points). Any problems not completed will be exempt. Note: these problems have well-established answers, so you can check your work.

Monty Hall Problem
You can read about the Monty Hall Problem here if you are not familiar:
https://en.wikipedia.org/wiki/Monty_Hall_problem

Write a Java program that simulates *n* "games" of Let's Make a Deal. Your program should track and report the total number of wins and losses after *n* games. Some stipulations:

- Your program should randomly generate the winning door each game.
- You may choose your initial contestant door however you would like.
- Allow your simulation to be run with switching doors allowed, and with switching doors NOT allowed. This way, you can very quickly show how win counts decrease when switching is not in play.

Birthday Problem
You can read about the Birthday Problem here if you are not familiar:
https://en.wikipedia.org/wiki/Birthday_problem

Write a Java program that simulates *n* "rooms" of randomly-chosen people. Your program should track and report the average number of people it takes to enter the room until a same-birthday is found. Some stipulations:

- For each guest, your program should randomly generate a birthday. You may do this with ints from 1 to 365.
- When a new birthday is generated, your program should check that it is different from all other already-generated birthdays in that room.
- If the randomly-generated birthday is the same as anyone currently in the room, the number of occupants gets factored into the average, and a new room begins.

Airplane Probability Problem
You can read about the Airplane Probability Problem here if you are not familiar:
https://medium.com/i-math/solving-an-advanced-probability-problem-with-virtually-no-math-5750707885f1

Write a Java program that simulates *n* "planes" of 100 people choosing their seats. Your program should track and report the number of times the final person ends up in their assigned seat. Some stipulations:

- You should randomly generate a seat number for the first person.
- Assume each subsequent person has their number as an assigned seat (person 2 gets seat 2, etc.).

- If a person's seat is taken when they enter the plane, randomly choose an untaken seat number for them. Otherwise, place them in their assigned seat.

Nontransitive Dice
You can read about Nontransitive Dice here if you are not familiar:
https://en.wikipedia.org/wiki/Nontransitive_dice

Write a Java program that simulates *n* "games" n which each of three dice goes head-to-head with each of the other two. One "game" consists of die A against B, then A against C, then B against C. Your program should track and report each die's record against each of the others. Some stipulations:

- Your program should assume the dice are labeled as follows:
    - Die A: 2, 2, 4, 4, 9, 9
    - Die B: 1, 1, 6, 6, 8, 8
    - Die C: 3, 3, 5, 5, 7, 7
- Randomly generate the "up" side of a dice each time it is rolled.

Penney's Game
You can read about Penney's Game here if you are not familiar:
https://en.wikipedia.org/wiki/Penney's_game

You will not only be simulating Penney's game, but also a few problems leading up to it who's answers may or may not surprise you. This is going to seem like a lot at first—once you get one of them though, most of the others are quite similar.

As a quick introduction, I think we can all agree that if two fair coins are flipped, we have an equal probability of the results HH and HT—both events have a 25% chance of occurring (our only four options are HH, HT, TH, and TT). Ok—no confusion yet—now the fun begins:

- Have your program simulate flipping a coin until you get two heads in a row (HH). Perform this *n* times, each time counting how many flips it takes. Do the same thing for *n* more games, but this time record the number of flips until you get heads followed by tails (HT). Average the two sets of flips (i.e., number of times until HH on average vs. number of times to HT on average). Are the answers the same?
- Ok—new game—have your program simulate flipping a coin repeatedly until HH emerges (meaning player 1 wins) or HT emerges (player 2 wins). Play this game *n* times. How often does each player win?
- Another new game—similar to the last one, but now we're looking for a sequence of three flips. Player 1 wins with HTT, player 2 wins with HHT. Play this game *n* times. How often does each player win?
- Final game—similar again, but now player 1 wins with HHH, and player 2 with THH. Play this game *n* times. How often does each player win?
- Please note—make your life easier and use Booleans! When I solved this, I also wrote separate methods for each "game" if you'd like to organize yourself that way—up to you!

Tuesday Birthday Problem
You can read about the Tuesday Birthday problem and those we'll explore leading up to it here if you are not familiar:

https://towardsdatascience.com/tuesday-birthday-problem-2927e83e5af3

You may have heard of part of this problem before, but probably not all of it. There are a few important assumptions. First, assume that each child is born male or female with equal probability of ½. Any individual child is independent of any other. A child is equally likely to be born any day of the week, and the day is independent of the sex.

- Have your program randomly generate *n* two-child families, essentially by randomly assigning a set of two children a sex and a day of the week born. Please note—you can perform the summations and conditionals below each time a new pair is generated—you don't have to store these pairs for the duration of the program.
- Count how many times a family's older child is male. Of these times, in how many is the younger child also male?
- Count how many times a family has at least one male child. In how many of these cases is the other child also male?
- Count how many times a family has at least one child a male born on a Tuesday. In how many of these cases is the other child also male (birthday does NOT matter for the second child).
- Remember here—you're doing simulations, not simply calculating probabilities—those aren't too hard for this particular question, but let's make our simulations prove them right!

Candies Problem
I can't find a concise online page for this one, so here is the setup:

You have a bag of candies in which there are 123 caramel candies and 321 mint candies. Every morning you randomly draw and eat candies from the pack until you get one that is different than each of that day's previous draws—when this happens, you return it to the back and wait for tomorrow. This means you're guaranteed to eat at least one candy a day, and may or may not eat several more after that. What is the probability that the last candy in the bag to be eaten will be a caramel?

Write a program that simulates *n* bags of candy with these conditions, and runs simulations on the eating process. Record the number of times the final candy in a bag is a caramel.

- You can use Boolean values instead of chars or Strings here to mimic the candies.
- You can implement the randomness in one of two ways—either have the candy bag ordered and randomly generate indices to pull from, or randomly order the bag.

100 Prisoners Problem
You can read about the 100 prisoners problem here if you are not familiar:
https://en.wikipedia.org/wiki/100_prisoners_problem

Write a program that simulates *n* sets of these 100 prisoners, and runs the method described in the "strategy" section of the link above each time. Record the number of times the entire set of 100 prisoners is pardoned, i.e., the number of times all 100 find their numbers in the drawers.