APMA 3100 Probability

Project 1: Counting Methods

This project will be done in groups of two.

Please turn in a document that has your responses to the prompts in each section below.

- Include the names of both members of the group.
- Please write and sign (or just type your name, if necessary) the honor pledge somewhere on the document.

For this project, you should not use any specialized functions or libraries (e.g. itertools) that your programming language might offer.

1. A student is registering for classes for next semester. The courses she can enroll in are calculus, physics, organic chemistry, history, literature, French, and astronomy. She has room for four classes in her schedule, at the following times: 9am, 10am, 11am, 12pm. She wants to count all the possible versions of her schedule. Additionally, the student's mother is helping her buy textbooks for her classes. Each course listed above has a specific textbook that is required with it. Therefore, a book order for the student will consist of four textbooks. Book orders are automatically listed in alphabetical order by the class subject (i.e. one book order could be astronomy, French, organic chemistry, physics). The student's mother wants to count all the possible book orders that she could be making.

    a. Make the beginnings of a carefully ordered list of all possible book orders. (Your list should include enough outcomes so that someone else can see and continue the pattern.)

    b. Now make the beginnings of a carefully ordered list of all the possible versions of the student's *schedule,* assuming she can't register one class for multiple slots. (Your list should include enough outcomes so that someone else can see and continue the pattern.

    c. Here is a program written in Python.

```python
orders = 0
books = ['calculus','physics','chemistry','history','literature','french','astronomy']
N = len(books)

for i in range(N):
    for j in range(N):
        if i != j:
            for k in range(N):
                if k != i and k != j:
                    for l in range(N):
                        if l != i and l != j and l != k:
                            order += 1
print(orders)
```

Here is the same program written in a slightly better way that takes advantage of Python's ability to iterate through lists:

```python
orders = 0
books = ['calculus','physics','chemistry','history','literature','french','astronomy']

for a in books:
    for b in books:
        if a != b:
            for c in books:
                if c != a and c != b:
                    for d in books:
                        if d != a and d != b and d != c:
                            orders += 1
print(orders)
```

Does this count the possible book orders, schedules, or neither? Briefly explain.

d. Write a program using for loops that *lists* AND *counts* all the possible book orders. Include a pasted copy of your code. It should display all the possibilities, and it should provide a final count.

Note 1: Consider using something like < or <= instead of !=.
Note 2: The logic and code for this problem and in problems 2 & 3 can and should closely resemble the logic and code provided in problem 1c.

2. Counting passwords…
   a. An 8-digit password is required to have three 0's and five 1's. You will determine how many unique passwords are possible.

   First consider how you might notate possible outcomes in the process of listing them. For example, any of the following can represent the same outcome:

          01011101
          137 (the digits are the locations of the 0's)
          24568 (the digits are the locations of the 1's)

   Write a program that lists and counts the unique passwords. Provide your code, the list of all outcomes, and the count.

   Note: A working program will give you most of the credit for this section. However, you'll get full credit if you write your program in such a way that you never actually have all of the passwords stored in an array or list or any other container object simultaneously, but rather you simply create each one, count it, and then forget it.

   b. Now suppose a 30-digit password is required to have thirteen 0's and seventeen 1's. You don't need to list all the possible passwords, but you need to determine how many exist. You may solve this however you wish, but briefly compare your method to what you used in part (a) and justify your chosen method.

3. An 8-digit password is required to have exactly three 0's. The other 5 digits can be any number 1-7, but numbers 1-7 may not be repeated.

   a. Again, consider different options for notating possible outcomes. (Hint: it may be easier to think of a way to notate the *location* of the 0's, rather than just writing the outcomes as a list of 8 digits) Then make a list of at least 20 of the possible passwords by hand. (List the outcomes in an ordered, intentional way so as not to miss any if you kept going.)

   b. Write a program that lists and counts all the possible outcomes. Provide a pasted copy of your code, the first 100 passwords that it produces, and the final count.

      Note 1: As in #2, you'll receive almost all of the credit here if your program works. But you'll get full credit if you write your program in such a way that you never actually have all of the passwords stored in an array or list or any other container object simultaneously, but rather you simply create each one, count it, and then forget it.
      Note 2: Your program may take a long time to run if it displays all of the possible passwords to the screen. Remember that you only need the first 100 displayed.

   c. Suppose you have a password with *m* digits required to have exactly *n* 0's. The other (m-n) digits can be any number 1-9 but numbers 1-9 may not be repeated. How many unique passwords exist? (Your answer should be a mathematical formula written in terms of m and n.)