We are going to get more practice with what we've learned about strings, lists, loops, conditionals and variables. This time, we are practicing on two questions that I have personally seen in interviews and one that is a simplified version of such a question.

Please make a copy of this Google Doc (File > Make a Copy) and fill in the answers below. Then, you can go to File > Download > pdf when you are done and **upload your answers to Canvas.**

Note: If you get stuck on this (or any) lab assignment, it is completely acceptable to 1) talk to a classmate and discuss what they think and 2) ask the instructor or CS tutor for help. See the syllabus for the appropriate email addresses if you are not doing this lab during the Thursday lab period. Here's a small hint if you are the type of person to carefully read the instructions: You do not need a while loop in any of the questions today. Attention to detail and taking in all the information you have are good skills to have.

1) [20 points] One application of the modulo operator is to detect errors in messages, like bar codes. Suppose we want to have a 10 digit code on our products, like UPC. We can have the first and last digit be "check digits" in that they will make sure the 10 digits in between were scanned or typed correctly.

Suppose we have the product code `3601283124`**. To compute the first and last check digit, we must add all the digits together after multiplying the second, fourth, sixth, eighth, and tenth digits by 5. In the example, we must compute**

```
3 + (6*5) + 0 + (1*5) + 2 + (8*5) + 3 + (1*5) + 2 + (4*5) == 110
```

The first check digit is this result of this sum modulo 7 and the last check digit is the result of this sum modulo 8. `110 % 7 == 5 and 110 % 8 == 6`**. Thus our first check digit is** `5` **and our last check digit is** `6`**. Therefore, we could write our final bar code[1] like:**

```
5 36012 83124 6
```

If you want to see why this is an error detecting code, try changing one of the middle 10 digits or transposing two of them - do both check digits still line up?

Let's make a program to help calculate those two check digits based on the above error detection algorithm. Your code should ask the user to type in a 6-14 digit number. Then, it should do the math outlined above and print out the first and last check digit.

Your solution should have at least one loop to handle any amount of product code digits. It should probably have at least one conditional. If you find yourself copying and pasting a lot of your code, you are probably doing something wrong. Here is the example output of what a successful program would show. The underlined part is what the user typed in:

```
Enter the product code: 3601283124
Here is the product code with the check digits:
5 3601283124 6
```

[1] Does this barcode look similar to any you've seen before? Grab a box of cereal or something to find one for comparison.

1a) [5 points] Write out the steps to implement the algorithm for question 1 and the Python concepts that will help achieve that. You do not necessarily need to fill every row, but the algorithm should have at least 4 steps. Fill this in <u>before</u> you start writing code. If you copy and paste your finished code in, you will receive 0 points and my ire.

Step	**Things we need**
1) Make a gatherer for the digits the user types in.	New variable. String holding all digits typed in num = {}
2) go through all the positions of the user's input	for range loop
3) Extract the digit at the current position and convert to an integer	Indexing int(
4)Ask if this position is even.	If statement If digits % 2 == 0
5)If it is, add the digit to the gatherer 1. Correct answer: 2. If it is not, add the digit times five to the gatherer	Else and elif statement else digits + 1 Elif digits
6)Calculate mod 7 and mod 8 of the gatherer	Num % 7 Num % 8
Display the two check digits and the original number to the user	print(Num)

1b) [15 points] Write the code to implement the algorithm from question 1

```
# Your code should go here
digits = input("Enter product code")
num = 0
for pos in range(0 , len(digits), 1):
    hop = digits[pos]
    hop = int(hop)
    if pos % 2 == 0:
        num += hop
    else:
```

```
        num += hop * 5
mod_7 = num % 7
mod_8 = num % 8
print("Total:", num)
print("Mod 7:", mod_7)
print("Mod 8:", mod_8)
```

2) [20 points] Write some code that prints out all the numbers from 100 to 200. However, if a number is evenly divisible by 4, instead of printing the number, print "bump ". If a number is evenly divisible by 5, instead of printing the number, print "set ". If a number is evenly divisible by 6, instead of printing the number, print "spike". Here is a video showing one potential solution.

2a) [2 points] If a number is evenly divisible by 4 and 5 (but not 6), it should print "bump set" instead of the number. Likewise, if a number is evenly divisible by 5 and 6 (but not 4), it should print "set spike" instead of the number. Give an example of a number in the provided range that is divisible by 4 and 5 (but not 6) and an example of a number in the provided range that is divisible by 5 and 6 (but not 4).
4&5= 100
5 & 6 = 150

2b) [1 point] If a number is evenly divisible by 4, 5, and 6, it should print "bump set spike" instead of the number. Give an example of such a number in the provided range.

4 & 5 & 6=120

2c) [2 points] Write out the steps to implement the algorithm for question 2 and the Python concepts that will help achieve that. You do not necessarily need to fill every row, but the algorithm should have at least 3 steps. Fill this in before you start writing code. If you copy and paste your finished code in, you will receive 0 points and my ire.

Step	**Things we need**
start a loop	For Range loop
see if number is divisible by 4	if statement
if not, see if number is divisible by 5 or 6	else statement
display end result	print(num)

2d) [2 point] What values is your stepper variable going to have? What order does it go through them?

num= integer , one at a time

2e) [13 points] Write the code to implement the algorithm from question 2

```
# Your code should go here
for num in range(100, 201):
    if num % 4 == 0:
        print("bump")
    elif num % 5 == 0:
        print("set")
    elif num % 6 == 0:
        print("spike")
    else:
        print(num)
```

3) [20 points] Suppose you are doing some data research to make a large language model, like GPT-3[2], which could be to make something like autocomplete. Part of that analysis requires knowing which words come after other words, for example, that "scenery" might come after "the", but "of" rarely does, so the computer doesn't recommend "of" after the user types the word "the".

Consider the following text:

Already it was deep summer on roadhouse roofs and in front of wayside garages, where new red gas-pumps sat out in pools of light, and when I reached my estate at West Egg I ran the car under its shed and sat for a while on an abandoned grass roller in the yard. The wind had blown off, leaving a loud bright night with wings beating in the trees and a persistent organ sound as the full bellows of the earth blew the frogs full of life. The silhouette of a moving cat wavered across the moonlight and turning my head to watch it I saw that I was not alone--fifty feet away a figure had emerged from the shadow of my neighbor's mansion and was standing with his hands in his pockets regarding the silver pepper of the stars. Something in his leisurely movements and the secure position of his feet upon the yard suggested that it was Mr. Gatsby himself, come out to determine what share was his of our local heavens.

[2] GPT-3 is a machine learning project which has processed a huge amount of text (hundreds of billions of words) and can now answer questions like a human might.

The following words all come after "the"
`[car, yard, wind, trees, full, earth, frogs, silhouette, moonlight, shadow, silver, stars, secure]`

Notice punctuation isn't a part of the words, nor are words repeated (even though yard shows up twice after though), and uppercase vs lowercase doesn't matter (both "The" and "the" were matched).

I want you to write some code that has the user paste in a bunch of text, and then type in a word (the target word). Your code should print out a list of all unique words, without punctuation, that come after the target word (not case sensitive) in the text. The target word in the above example was "the".

3a) [3 points] If the target word was "it", what list of words should your code print out from the example text?
was
I
was

3b) [optional] This space is for you to plan your work. You may get partial credit based on your algorithm and planning here if your code does not work fully.

3c) [17 points] Write the code to complete question 3.

```
# Your code should go here
import string
def main():
    text = input("Please paste your text here: ")
    target_word = "the"

    cleaned_text = text.replace("." , "") and text.replace("," , "")
    words = cleaned_text.split()

    following_words = []

    for i in range(len(words) - 1):
        if words[i].lower() == target_word:
            following_words.append(words[i + 1])

     if following_words:
        print("Words following 'the':")
        for word in following_words:
            print(word)
    else:
        print("There are no words following 'the' in the text.")
```

```
if __name__ == "__main__":
    main()
```

4 [bonus; 10 points] Look back at the error detection problem from question 1. Can you find any cases where a single mistyped digit (e.g. mistyped a 7 instead of a 1) or transposed set of adjacent digits (63 instead of 36) would not be detected correctly? The way errors are detected are noticing the check digits have changed, so a "silent error" would be a typo that results in the not changing the check digits. Explain how you used code or mathematical rigor to find such errors or prove they cannot occur.