CS 100 Scientific Computing Midterm Exam 2 November 11, 2022

1.) Strings (20 pts.)

Give a series of Python commands that creates the result string from the original string:

a.)
```
original =    "the big bad bunny bites bats"
result =      "the zig zad zunny zites zats"

result = ""
for letter in original:
   if letter == "b":
         result += "z"
   else:
         result += letter
```

b.)
```
original =    "abcdefg"
result =      "bcdefgh"

result = ""
for letter in original:
         num = getAlphaPos(letter) # function from class
         result = result + chr(((num+1) % 26) + ord('a'))
```

c.)
```
original =    "abcabcabcabc"
result =      "abababab"

result = ""
for letter in original:
   if letter != "c":
         result += letter
```

d.)
```
original =    "aeiouy"
result =      "aaeeiioouuyy"

result = ""
for letter in original:
         result = result + letter + letter
```

e.)
```
original =    "potatopotato"
result =      "opatotopatot". #swaps pairs of letters

result = ""
for idx in range((len(original)//2)):
    result = result + original[(idx*2)+1]+original[idx*2]

if len(original)%2==1:
   result += original[-1]
```

2.) Understanding Code (20 pts.)

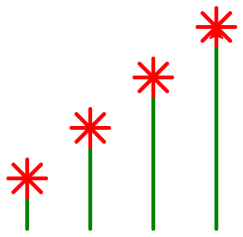
Show the output of this program when it is run with the command below. (Show what is drawn on the Canvas)

```
import turtle

t = turtle.Turtle()
location = -50
distance = 40
t.width(3)
for num in range(4):
    t.penup()
    t.goto(location, -50)
    location += 50
    t.pendown()
    t.color("green")
    t.setheading(90)
    t.forward(distance)
    distance += 40
    for val in range (8):
        t.color("red")
        t.forward(15)
        t.backward(15)
        t.right(45)
```

Note to those practicing for the exam:
To best prepare, do not simply type this program into Thonny and see the results. Walk through the steps of the code and demonstrate to yourselves that you understand the program well enough to produce the same output as Thonny.

If you enter this program into Thonny, you get a drawing of four plants increasing in height to the right. Each plant has a red "flower" with 8 petals sticking straight out from the center.



3.) Loops: (20 pts.)
For each of the loops below, write another loop that produces the same output. If the loop given is a FOR loop, write a WHILE loop to do the same thing. If the loop given is a WHILE loop, write a FOR loop to do the same thing. If it is not possible to write an equivalent loop, explain why.

a.) Loop 1

```
value = 0
for num in range(20):
      if num % 2 == 0:
            value = value + num
      else:
            value = value + 1
print(value)
```

```
value = 0
num = 0
while num < 20:
      if num % 2 == 0:
            value += num
      else:
            value += 1
print(value)
```

b.) Loop 2

```
num = 0
while num < 10:
      val = random.randint(0,9)
      print(val, end=" ")
      num += 1
print()
```

```
for num in range(10):
      val = random.randint(0,9)
      print(val, end=" ")
      num += 1
print()
```

c.) Loop 3

```
num = random.randint(0,9)
while num > 0:
      num = random.randint(0,9)
      print(num, end=" ")
print()
```

Not possible, the number of iterations required is not known, so a WHILE loop MUST be used.

d.) Loop 4

```
for num in range(1,50,5):
      print(num, end="")
      if num > 25:
            print("x", end=" "
      else:
            print("o", end=" "
print()
```

```
num = 1
while num < 50:
      print(num, end="")
if num > 25:
      print("x", end=" ")
else:
      print("o", end=" ")
num += 5
print()
```

e.) Loop 5

```
count = 0
while count < numValues:
 num = random.randint(0,99)
 if num % 2 == 1:
   print("All values not even!")
   return count
 count += 1
print("All numbers were even"))
return count
```

```
for count in range(numValues):
    num = random.randint(0,99)
    if num % 2 == 1:
       print("NOT ALL even")
       return count
print("All number were even")
return count
```

4.) Programming: (20 pts.)

Write a function called atLeastThree() that takes a list as a parameter and prints out all the values in the list that occur at least three times. IMPORTANT: The lists passed in will ONLY contain values 0 through 9. (This task is much more difficult without this restriction) See examples below:

```
numList1 = [1,2,3,4,5,6,7,8,9,5,0,5,4]

numList2 = [1,2,3,1,2,3,1,2,3,0,0,0,4,5,6]
numList3 = [7,7,7,7,7,7,7,7,7,7,7,7,7]
numList4 = [0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,2,8]

atLeastThree(numList1)
5
atLeastThree(numList2)
0 1 2 3
atLeastThree(numList3)
7
atLeastThree(numList4)
2 8
```

```
# Uses a nested for loop to compare all values in list to 0-9
def atLeastThreeV1(numList):
      for match in range(10):
            count = 0
            for num in numList:
                  if match == num:
                        count += 1
            if count >= 3:
                  print(match,end=" ")
      print()
```

```
#Uses a list to count how many of each digit
def atLeastThreeV2(numList):
      digitCount=[0,0,0,0,0,0,0,0,0,0]
      for num in numList:
            digitCount[num] += 1

      for idx in range(10):
            if digitCount[idx] >= 3:
                  print(idx,end=" ")
      print()
```

5.) Short Answer: (20 pts.)

Answer each of the following with a SHORT answer – no more than TWO sentences.

a.) State the major difference between strings and lists.

Strings are immutable (cannot be changed), lists can be modified

b.) Write an equivalent Python command to this: **`num += val`**

`num = num + val`

c.) if num is 0 and count is 5, how many times does this while loop iterate? Explain

```
while num >= 0 and count < 10:
     num = num + 1
     count = count + 1
     print(num, " ", count)
```

This loop will iterate 5 times, when count is 5,6,7,8, 9. When count becomes 10, it is no longer less than 10, so the while loop Boolean becomes false and the loop stops.

d.) If num is 0 and count is 5, how many times does this while loop iterate? Explain

```
while num >= 0 or count < 10:
     num = num + 1
     count = count + 1
     print(num, " ", count)
```

This is an inifinte loop that will never stop. Yes, count will become 10 on the fifth iteration and count<10 will no longer be true, but since the Boolean expression uses an "or" and the first clause "num >= 0" will always be true, the whole expression will never be false. So the loop will not end.

e.) Suppose we have the following list and function. What is the output when the command `isThereAFive(numbers)` is executed? Explain.

```
numbers = [0,1,2,3,4,5,6]
def isThereAFive(numList):
     for num in numbers:
          if num == 5:
               return True
          else:
               return False
```

This will produce the INCORRECT answer of False. All values must be looked at to determine if something is not there. The should be no else clause; return False should be outside the loop.

f.) What is produced from this operation: Explain

```
chr(5 + ord('a') )
```

```
e
```

g.) What is produced from this operation: Explain

```
(ord('y') + 5) % ord('z')  # changed question
```

```
4
```

h.) Explain the difference between ending a function with `print(result)` and ending a function with `return result`.

The difference between "print" and "return" is that print simply reports a value to the console so that a human can read it. Nothing else happens. The return does not put anything on the console, but it delivers the resulting value to the portion of the code that called the function and that value can be used in subsequent calculations.

i.) Suppose you want to swap two values in a list, so you write the following function. Will this correctly perform the swap? Explain.

```
def swapListValues(numbers, loc1, loc2):
    numbers[loc1] = numbers[loc2]
    numbers[loc2] = numbers[loc1]
```

No, swaps require a temporary variable or a simultaneous operation.

```
temp = numbers[loc1]
numbers[loc1] = numbers[loc2]
numbers[loc2] = temp

-or-

(numbers[loc1],numbers[loc2])=(numbers[loc2],numbers[loc1]
```

j.) Explain the difference between **`True`** and **`"True"`**

`True` is the value of a Boolean variable that indicates the logical value representing a Boolean expression that is true (i.e. 7 > 3).

`"True"` is the string consisting of the letters 'T', 'r', 'u', and 'e'. It has nothing to do with Boolean expressions.