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”
```

b.)
```
original =    “abcdefg”
result =      “bcdefgh”
```

c.)
```
original =    “abcabcabcabc”
result =      “abababab”
```

d.)
```
original =    “aeiouy”
result =      “aaeeiioouuyy”
```

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

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.

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)
```

b.) Loop 2

```
num = 0
while num < 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()
```

d.) Loop 4

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

e.) Loop 5

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

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.

b.) Write an equivalent Python command to this: **`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)
```

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)
```

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
```

f.) What is produced from this operation: Explain

```
     chr(5 + ord(‘a’) )
```

g.) What is produced from this operation: Explain

```
     (ord(‘y’) + 5) % 26
```

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

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]
```

j.) Explain the difference between **`True`** and **“`True`”**