Question 1

def oddMod3(lst, modulus):
    for num in lst:
        if num%2 == 0:
            continue
        if num%modulus != 3:
            return False
    return True

numList = [0, 8, 13, 2, 7]
print(oddMod3(numList, 5))
a. True
b. False
c. True False
d. SyntaxError: multiple return statements
e. None of the above

Question 2

def initialVowel(words):
    vowels = 'aeiou'
    rtnList = []
    for word in words:
        if word[0].lower() in vowels:
            rtnList.append(word)
        return rtnList

wordList = ['At', 'the', 'Open', 'many', 'balls', 'in', 'play']
print(initialVowel(wordList))
Question 3

t = 'How much wood would a woodchuck chuck if a woodchuck could chuck wood?'

woodCount = t.count('wood')

chuckCount = t.count('chuck')

print(woodCount, chuckCount)

a. 2 2
b. 2, 2
c. 4 4
d. 4, 4
e. None of the above

Question 4

wolfLine = "Let me in let me in or I'll huff and I'll puff and I'll blow your house in"

wolfWords = wolfLine.split()

commonist = wolfWords[0]

comCount = wolfWords.count(wolfWords[0])

for word in wolfWords:
    wordCount = wolfWords.count(word)
    if wordCount > comCount:
        commonist = word
        comCount = wordCount

print(commonist)

a.

b. in
Question 5

def sWords(strList):
    sWords = []
    for word in strList:
        if word[len(word)-1] in 'sS' and word not in sWords:
            sWords.append(word)
    return sWords

s = ['Sweet', 'dreams', 'be', 'yours', 'dear', 'If', 'dreams', 'there', 'be']
print(sWords(s))

a. ['Sweet', 'dreams', 'yours', 'dreams']
b. ['Sweet', 'dreams', 'yours']
c. ['dreams', 'yours', 'dreams']
d. ['dreams', 'yours']
e. None of the above

Question 6

musicStuff = [['Woody', 'Dylan'], 'Ludwig', {'band': ['John', 'George', 'Paul', 'Ringo']}, 'Clash']
print((musicStuff[2])['band'])

a. TypeError: list indices must be integers, not str
b. ['John', 'George', 'Paul', 'Ringo']
c. 'Ludwig'
d. 'Clash'
e. None of the above

Question 7
jingle = {1:'money', 2:'show', 3:'get ready', 4:'go'}

print(jingle[0])

a. 'money'
b. 1:'money'
c. IndexError: index out of range
d. KeyError: 0
e. None of the above

**Question 8**

boolExprs = [True, not False, False and not True, not False or True]

falseCount = 0

for expr in boolExprs:
    if expr == False:
        falseCount += 1

print(falseCount)

a. 1
b. 2
c. 3
d. 4
e. None of the above

**Question 9**

The lines below are the content of the file named 'humpty.txt'.

*Humpty Dumpty sat on a wall,*
*Humpty Dumpty had a great fall.*
*All the king’s horses and all the king’s men*
*Couldn’t put Humpty together again.*

After the execution of the following code, what is the content of the file 'humptyOut.txt'?

def odds(inName, outName):
    inF = open(inName, 'r')
    outF = open(outName, 'w')
for text in inF:
    lst = text.split()
    if len(lst) % 2 == 1:
        outF.write(text)

inF.close()
outF.close()

print(odds('humpty.txt', 'humptyOut.txt'))

a. Humpty Dumpty sat on a wall, Humpty Dumpty had a great fall.
b. Humpty Dumpty sat on a wall, All the king's horses and all the king's men
c. Humpty Dumpty had a great fall. Couldn't put Humpty together again.
d. All the king's horses and all the king's men Couldn't put Humpty together again.
e. none of the above

Question 10

import turtle

t = turtle.Turtle()

num = 90

for i in range(1, 5):
    t.forward(num)
    t.right(num * i)

a. A square
b. A triangle
c. Two steps of a staircase
d. A straight line of length 360
e. None of the above

Question 11

i = 1

while i < 4:
if i%2 == 0:
    i /= 2
else:
    i *= 3
i += 1
print(i)

a. 1
b. 2
c. 3
d. 4
e. none of the above

**Question 12**

refrain = 'hallelujah'

print(refrain[:5] + refrain[5:])

a. halleelujah
b. hallellujah
c. halleujah
d. hallelujah
e. None of the above

**Question 13**

def fTest(thing):
    if len(thing)%2 == 0:
        return "even"
    else:
        return "long"
x = []

print(fTest(x))

a. even
b. short
c. even short
d. Syntax error: hanging 'else'
e. None of the above

**Question 14 Part a (6 points)**

Write a function named `drawLine()` that takes two parameters:

1. `t` – a turtle used for drawing and
2. `length` – the length of the line to draw

The function `drawLine()` should draw a line of the specified length. It should leave the turtle `t` in the initial position and orientation.

**Question 14 Part b (14 points)**

Write a function named `spikes()` that draws lines radiating from a common starting point. The function `spikes()` takes three parameters (each a non-negative integer):

1. `numLines` – the number of lines to draw
2. `lengthIncr` – the length of the first line and the increase in length of successive lines
3. `angle` – the (clockwise) angle between successive lines

The function `spikes()` should call `drawline()` repeatedly. `spikes()` should create a turtle and pass it to `drawLine()` as a parameter.

For example, the function call

```
spikes(36, 25, 5)
```

should produce the following output
Question 15 (20 points)

Write a function named newWords() that, for each line of an input file i) identifies the words that have not occurred in any prior line of the input file and ii) writes the set of new words (without duplication) to a new line in the output file.

The function newWords() takes two parameters:

1. inFile - a string that is the name of an existing text file
2. outFile - a string that is the name of a file to write the function output

You may assume that the words in the input file are separated by spaces and consist only of lower case characters, with no punctuation. For example, suppose that the file directions.txt contains these lines

you have brains in your head
you have feet in your shoes
you can steer yourself any direction you choose

Then the following function call

newWords('directions.txt', 'directionsNew.txt')

should produce an output file named directionsNew.txt with this content

you have brains in your head
feet shoes
can steer yourself any direction choose

Question 16a (7 points)

Write a function named uniqueLetterCount() that takes a string as a parameter and returns the number of different letters in the string. You may assume that the string contains only lower case letters. For
example, if the parameter is the string ‘hurrah’, uniqueLetterCount() should return 4 (because the unique letters are ‘h’, ‘u’, ‘r’, and ‘a’.

Questions 16b (13 points)

Write a function named uniqueLetterDict() that takes a single parameter t (a string). The string t contains zero or more words separated by spaces, composed only of lower case letters. The function uniqueLetterDict() should return a dictionary in which the keys are the number of unique letters in the words of t. For each key, the corresponding value is a list of all the words in t having exactly that many unique letters.

For example, the function uniqueLetterDict() is called by these lines

paranoids = 'even paranoids have real enemies'
print(uniqueLetterDict(paranoids))

the following would be correct output

{8: ['paranoids'], 3: ['even'], 4: ['have', 'real'], 5: ['enemies']}