**Question 1**

```python
start = 7
for i in range(3, 0, -1):
    start += i
print(start, end = ' ')
```

a. 8 10 13  
b. 7 8 10  
c. 6 4 1  
d. 10 12 13  
e. none of the above

**Question 2**

```python
dog = 'Zoomie'
pattern = ''
for letter in dog:
inPattern = False
    if dog.count(letter) > 1:
inPattern = True
        pattern += letter
    if dog.count(letter) > 0 and inPattern == False:
inPattern = True
        pattern += letter
    else:
inPattern = False
        pattern += '.
print(pattern)
```

a. SyntaxError: invalid syntax  
b. Zoomie  
c. Zo.o.mie  
d. Zo.mie  
e. none of the above

**Question 3**

```python
magic = 'abracadabra'
print(magic[:4] + magic[-4:] + magic[4:11])
```

a. IndexError: string index out of range  
b. SyntaxError: invalid syntax  
c. ababraabracadabra  
d. abraabraabracadabra  
e. none of the above
Question 4

vowelCombos = ['oo', 'ei', 'ie', 'ai', 'ea', 'io']
nonsense = 'each to their own idiom, and to all a good night'
combos = []
for combo in vowelCombos:
    if combo in nonsense:
        combos.append(combo)

print(combos)

a. ooeieaio
b. ['oo', 'ei', 'ea', 'io']
c. ['oo', 'ei', 'ie', 'ai', 'ea', 'io']
d. TypeError: Can't convert 'seq' object to str implicitly
e. none of the above

Question 5

def rareLetterCheck(testStr):
    rareLetters = ['j', 'k', 'q', 'x', 'z']
    for letter in testStr:
        if letter in rareLetters:
            return letter

s = 'Quick brown fox'
print(rareLetterCheck(s))

a. Q
b. qkx
c. Qkx
d. x
e. none of the above

Question 6

mixedTypes = [['Batman', 'Robin'], [0, 1], [3.14, 2.17]]
print(mixedTypes[1:2])

a. [0, 1]
b. ['Batman', 'Robin'], [0, 1]
c. [0, 1], [3.14, 2.17]
d. TypeError: incompatible types
e. none of the above
Question 7

```python
>>> costumes = [['ghost', ['spider']], ['ghoul', ['zombie', 'pirate']], ['vampire', ['devil']]]
>>> print(costumes[1][1])
```

a. ['ghost']
b. ['ghost', ['spider']]
c. ghoul
d. ['zombie', 'pirate']
e. none of the above

Question 8

```python
def reps(aStr, subStrs, times):
    frequent = []
    for subStr in subStrs:
        if aStr.count(subStr) >= times:
            frequent.append(subStr)

    return frequent

>>> text = 'Make that cat go away! Tell that Cat in the Hat you do NOT want to play.'
>>> patterns = ['that', 'cat', 'ay']
>>> print(reps(text, patterns, 2))
```

a. TypeError: incompatible data types 'subStrs'
b. 'that', 'ay'
c. ['that', 'ay']
d. None
e. none of the above
**Question 9**

```python
boolExprs = [True and False, True or False, True, False]
print(trueCount(boolExprs))

def trueCount(boolList):
    trueCount = 0
    for expr in boolList:
        if expr == True:
            trueCount += 1
    return trueCount
```

a. 0  
b. 1  
c. 2  
d. 3  
e. none of the above

**Question 10**

```python
def testNest(aStr, subStr):
    for i in range(len(aStr)):
        for j in range(len(aStr[i:])):
            if aStr[i:j] == subStr:
                return [i, j]
s0 = 'viper'
s1 = 'vip'
print(testNest(s0, s1))
```

a. None  
b. [0, 2]  
c. [0, 3]  
d. ['viper', 'vip']  
e. none of the above
Question 11

```python
names = open('names.txt', 'w')
names.write('Joe' + '
')
names.write('Mary' + '
')
names.close()
names = open('names.txt', 'r')
print(names.read())
names.close()
```

a. JoeMary
b. Joe
   Mary
c. NameError: name 'names' is already defined
d. NameError: name 'names' is not defined
e. none of the above

Question 12

```python
def uniqueWords(phrase):
    unique = []
    words = phrase.split()
    for word in words:
        if words.count(word) == 1:
            unique.append(word)
    return unique

popEye = "I am what I am"
print(uniqueWords(popEye))
```

a. 'what'
b. [what]
c. ['what']
d. TypeError: uniqueWords() takes 1 argument but 5 were given
e. none of the above
Question 13 (20 points)

Write a function named droplets() that uses turtle graphics to draw a series of droplets (circles). (Hint: there is a turtle method circle that draws a circle of a specified radius.) The droplets all fall along a straight line. The droplets are separated by distances that are specified in the function parameter separations.

The function droplets takes three parameters:

i. a turtle, t, that is used to draw the droplets
ii. an int, size, that is the diameter of each droplet
iii. a list of int's, separations, that is the distance between corresponding points of successive droplets.

Note 1: The number of droplets drawn is one greater than the number of separations. For example, a list of three separations implies that there are four droplets. Note 2: The question does not tell you anything about the up/down or other state of the turtle that is passed to the function, and it does not require that the turtle be left in any particular state.

For example, the following would be correct graphical output:

```python
import turtle
turt = turtle.Turtle()
droplets(turt, 5, [20, 25, 30])
```

Question 14 (20 points)

We define the letters 'a', 'e', 'i', 'o' and 'u' as vowels. We do not consider any other letter as a vowel.

Write a function named mostlyVowels() that returns a list of words in a body of text in which more than half the letters are vowels. Count both capitalized and lower case instances of the vowels.

A word should appear in the return list at most once, no matter how many times it occurs in the input string. A capitalized instance of a word is the same word as a lower case instance. For example, 'Is' and 'is' are the same word.

Input: a string s that consists of words, separated by spaces
Return: a list of all words in the input string in which more than half the letters are vowels

For example, the following would be correct output:

```python
>>> mlk = 'Our lives begin to end the day we become silent about things that matter'
>>> print(mostlyVowels(mlk))
['our', 'about']
```
Question 15 (20 points)

Write a function named duplicateWordLines() that identifies lines in a file that have duplicate words and writes them to a new file.

The function duplicateWordLines() takes two string parameters. The first parameter is the name of an input file that exists before duplicateWordLines() is called. The second parameter is the name of an output file that duplicateWordLines() creates and writes to.

You may assume that the input file is in the current working directory and you should write the output file to that directory as well.

For each line in the input file, the function duplicateWordLines() should write to the output file only those lines that contain more than one instance of some word other than an article. Articles are the words 'a', 'an' and 'the'.

Assume that all punctuation has been removed from the input text.

For example, if the following is the content of the file bells.txt:

Keeping time time time
In a sort of Runic rhyme
To the throbbing of the bells
Of the bells bells bells

The following function call:

inF = 'bells.txt'
outF = 'bellsRepLines.txt'
duplicateWordLines(inF, outF)

should create the file ‘bellsRepLines.txt’ with the content:

Keeping time time time
Of the bells bells bells