Multiple choice questions 1-10 (4 points each)

Question 1

```
boolsSeen = 0
bools = [not True, not False, True, False, True and False, True or False]
for expr in bools:
    if expr:
        boolsSeen += 1
print(boolsSeen)

# Hint: x += 1 is the same as x = x+1
```

a. SyntaxError: invalid syntax
b. 1
c. 2
d. 3
e. none of the above

Question 2

```
aSeq = [2, 1, 0, -1, -2]
sum = aSeq[0] + aSeq[-1] + aSeq[-2]
print(sum)
```

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

Question 3

```
mix = ['zero', 0, ['two'], -1]
print(mix[0:-1])
```

a. [0, ['two']]
b. [0, ['two'], -1]
c. ['zero', 0, ['two']]
d. ['zero', 0, ['two'], -1]
e. none of the above

Question 4

```
aList = ['one', -1, 2]
prefix = aList[:2]
suffix = aList[-1:]
print(prefix + suffix)
```

a. ['one', -1, 2, -1, 2]
b. ['one', -1, -1, 2]
c. 1
d. [2, -1]
e. none of the above

Question 5
```
import turtle
s = turtle.Screen()
t = turtle.Turtle()
for i in range(4):
    if i%2 == 0:
        t.right(60)
        t.forward(100)
        t.right(60)
```

a. a straight line
b. two sides of a square
c. two sides of an equilateral triangle
d. an equilateral triangle
e. none of the above

Question 6
```
def check(aList):
    for element in range(len(aList)):
        if str(aList[element]) == aList[element+1]:
            return True
    return False

arg = [0, '0', 1, '1']
matched = check(arg)
print(matched)
```

a. True
b. False
c. True True
d. True False True
e. none of the above

Question 7
```
muchSnow = False
veryCold = True
takeTrain = True

if muchSnow:
    print("school closed")
else:
    print("give exam")
if veryCold:
    print("car won't start")
elif takeTrain:
    print("take exam")
else:
    print("miss exam")
```

a. give exam
b. give exam
   car won't start
   take train
c. give exam
   car won't start
   miss exam
d. SyntaxError: multiple 'if'
e. none of the above

Question 8
```
isaac = ['I','do','not','fear','computers','I','fear','the','lack','of','them']
short = 3
shortCount = 0

for word in isaac:
    if len(word) <= short:
        shortCount += 1

print(shortCount)
```

a. 0
b. 2
c) 4
d) 6
e) none of the above

Question 9
```
def notIn(letter, wordList):
    rtnList = []
    for word in wordList:
        if letter not in word:
            rtnList.append(word)
    return rtnList

quote = ['round', 'up', 'the', 'usual', 'suspects']
print(notIn('e', quote))
```

a. ['round']
b. ['round', 'up', 'usual']
c. []
d. TypeError: argument of type 'int' is not iterable
e. none of the above

Question 10
```
def accumulate(sequence):
    returnVal = []
    for element in sequence:
        returnVal.append(element)

    return returnVal

print(accumulate('anagram'))
```

a) ['anagram']
b) ['angrm']
c) 'anagram'
d) ['a','n','a','g','r','a','m']
e) none of the above

Programming questions 11-13 (20 points each)

Question 11

Part A: 10 points

A tick is a short line that is used to mark off units of distance along a line.

Write a function named drawTick() that uses a turtle parameter to draw a single tick of specified length perpendicular to the initial orientation of the turtle.

The function drawTick() takes two parameters:

1. a turtle, *t*, that is used to draw
2. an integer, *tickLen*, that is the length of the tick

When drawTick() is called, *t* is in the location that the first tick should be drawn. (Hint: remember that the tick mark should be drawn perpendicular to the orientation that *t* is in when it is called.)

You should not make any assumptions about the initial up/down state of *t*.

On return from drawTicks(), *t* should be in the same location and have the same orientation that it had when it was called.

Part B: 10 points

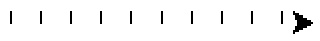
Write a function named drawTicks() that calls drawTick() repeatedly to draw a set of parallel tick marks.

The function drawTicks() takes four parameters:

1. a turtle, *t*, that is used to draw
2. an integer, *tickLen*, that is the length of the tick
3. an integer, *numTicks*, that is the number of ticks to draw
4. an integer, *distance*, that is the distance between parallel tick marks

For example, the following would be correct output if drawTicks() were called by this code:

```
import turtle
s = turtle.Screen()
turt = turtle.Turtle()
drawTicks(turt, 5, 10, 15)
```



Question 12 (20 points)
Write a function named beginsWith() that computes how many strings in a list of strings begin with a specified letter.

The function beginsWith() takes two parameters:

1. *letter*, a string of length 1
2. *strList*, a list of 0 or more strings

The function beginsWith() should return an integer that is the number of strings in *strList* that begin with *letter.*

You may assume that no word in *strList* begins with a capital letter.

The following is an example of correct input and output for the function beginsWith():

```
>>> eliza = ['the','rain','in','spain','falls','mainly','on','the','plain']
>>> firstLetter = 't'
>>> print(beginsWith(firstLetter, eliza))
2
```

Question 13
Write a function named greeting(). The function greeting() should ask the user for their name, and then ask the user for the day of the week. It should then greet the person by name and day and comment whether their name has fewer, more than or the same number of characters as the day.

The function greeting() takes one parameter: a string named *greetStr.*

The following is an example of correct input and output for the function greeting():

```
>>> greeting('Happy')
What's your name? Justin
What day is today? Monday
Happy Monday Justin
Your name has the same number of characters as today!
```