Question 1

```
value = 0
if True or False:
    value += 1
elif not True or not False:
    value += 2
else:
    value += 4
if True and False:
    value += 8
elif True:
    value += 16
print(value)
```

Hint: `value += 1` is the same as `value = value+1`

a) 13
b) 14
c) 18
d) 10
e) none of the above

Question 2

```
aList = [-1, 0, 1]
part1 = aList[:1]
part2 = aList[-3:]
print(part1 + part2)
```

a) [-1, 0, 1]
b) [-1, -1, 0, 1]
c) IndexError: list index out of range
d) [-1, 0, -1, 0, 1]

none of the above

Question 3

```
nested = ['', 0, ['']]
print(nested[1:2])
```

a) [0, ['']]
b) 0
c) [0]
d) ['', 0]
e) none of the above

Question 4

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

a) 1
b) -1
c) Index Error: list index out of range
d) 0
e) none of the above

Question 5

```
import turtle
s = turtle.Screen()
t = turtle.Turtle()
ints = [6, 5, 4, 3, 2]
for i in ints:
    if i%2 == 1:
        t.forward(100)
        t.right(90)
```

a) three sides of a square
b) a square
c) a straight line
d) two adjacent sides of a square
e) none of the above

Question 6

```
def buildStr(s):
    mid = ""
    end = ""
    for letter in s:
        mid = letter + mid + letter
        end = letter + end
    return s + mid + end

newString = buildStr("ab")
print(newString)
```

a) abbaabba
b) ababbaba
c) ababbaab
d) abbaabab
e) none of the above

Question 7

```
highWinds = True
highTide = True

if highTide:
    print('three feet of water')
if not highWinds:
    print('dodged a hurricane')
elif highTide and highWinds:
    print('Sandy!')
else:
    print('NJ is saved')
```

a) three feet of water
   NJ is saved
b) three feet of water
   dodged a hurricane
c) SyntaxError: multiple 'if'
d) three feet of water
   Sandy!
e) e. none of the above

Question 8

```
star = 'miss piggy'

myStr = ''
for i in range(len(star)-1):
    if star[i] != star[i+1]:
        myStr += star[i]

print(myStr)
```

a) mispig
b) mis pig
c) miss pigg
d) sg
e) none of the above

Question 9

```
def compare(thing1, thing2):
    if thing1 < thing2:
        return 'first is less'
    if not (thing1 == thing2):
        return 'not equal'
    if thing1 > thing2:
        return 'first is more'

comparison = compare('A', 'B')
print(comparison)
```

a) first is less
   not equal
b) ['first is less', 'not equal']
c) SyntaxError: multiple return statements
d) first is less
e) none of the above

Question 10

```
def mirror(aString):
    rtn = ""
    for i in range(len(aString)):
        if aString[i] == aString[-i-1]:
            rtn = aString[i] + rtn + aString[-i-1]

    return(rtn)

print(mirror('nabob'))
```

a) bob
b) b
c) bb
d) naob
e) none of the above

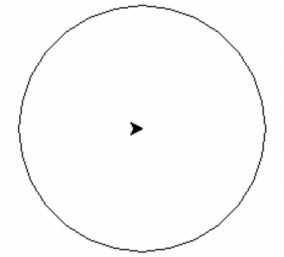
Question 11A (10 points)

Write a function named *concentric* that uses turtle graphics to draw a concentric (same center) circle of specified radius. The function *concentric* takes two parameters:

1. *t*, a turtle that is used to draw the circle. The turtle *t* may be in any position, orientation and up/down state.
2. *radius*, a positive integer that is the radius of the circle to draw

The function *concentric* should:

1. draw a circle whose center is the initial position of *t*
2. leave *t* in its initial position and orientation when it returns

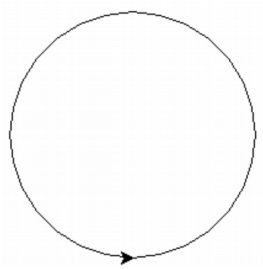


For example, if *aTurtle* is pointed to the right, the function call

```
concentric(aTurtle, 100)
```

should produce the graphical output at the left.

The function *concentric* may use the *circle* method in the turtle class, which draws a circle of specified radius in a counterclockwise direction. ***Note: the circle method draws a circle with the turtle on the perimeter, not at the center.*** For example, `t.circle(100)` draws the circle at the right.



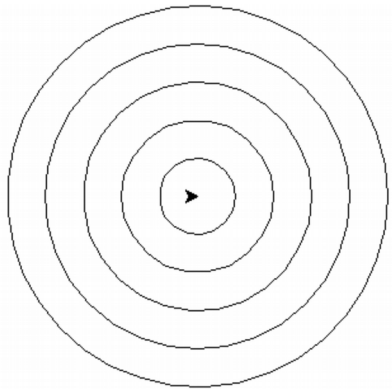
Question 11B (10 points)

Write a function named *dartboard* that uses the *concentric* function of Question 11A to draw a set of concentric circles of increasing radius. Call *concentric* and assume that it works correctly, even if you are not sure that your solution to Question 11A is completely correct.

The function *dartboard* takes three parameters:

1. *turt*, a turtle that is used to draw the concentric circles
2. *numRings*, an integer that is the number of circles to draw
3. *delta*, a positive integer that is the radius of the first circle and also the increase in the radius of each successive circle

The function *dartboard* should leave *turt* in its initial position and orientation. For example, the function call `dartboard(aTurt, 5, 20)` should produce the following graphical output.



Question 12 (20 points)
Write a function name *strLenParity* that computes and returns how many strings in a list of strings are of even length and how many are odd. The function *strLenParity* takes a single parameter, *stringList*, that is a list of strings. It should return a list of two integers, the first being the number of even length strings in *stringList* and the second being the number of odd length strings in *stringList*. For example, the following would be correct output.

```
singles = ['one','two','three','four','five','six','seven','eight','nine']
print(strLenParity(singles))
[3, 6]
```

Question 13 (20 points)
Write a function named *nameLenComment*. The function *nameLenComment* should prompt the user for a name and then print a comment telling the user whether their name is short, about average, or long. (You may assume that the user provides valid input.)

The function *nameLenComment* takes two integer parameters: *short* and *long*. If the length of the user's name is less than or equal to *short*, print the user's name and the comment, “your name is short”. If the length of the user's name is greater than or equal to *long*, print the user's name and the comment, “your name is long”. Otherwise print the user's name and the comment, “your name is about average length”.

The function *nameLenComment* should return the user's name.

For example, the following would be valid input and output.

```
print(nameLenComment(4,8))
What's your name? Athena
Athena, your name is about average length
Athena
```