Multiple choice questions 1-10 are worth 4 points each

Question 1

```
x = 2
for i in range(3):
    print(x, end = ' ')
    x += i
```

a. 0 1 2
b. 2 3 5
c. 2 2 3
d. 1 2 4
e. none of the above

Question 2

```
noise = 'hullaballoo'
idx = 0
while idx < len(noise):
    let = noise[idx]
    print(let, end='')
    letCount = noise.count(let)
    if idx > 2:
        idx += letCount
    else:
        idx += 1
```

a. an infinite sequence of h's
b. hulal
c. hulll
d. hlaal
e. none of the above

Question 3

```
s = 'GimmeASlice'
print(2*s[:5] + s[:])
```

a. GimmeGimmeGimmeASlice
b. GimmeAGimmeAGimmeASlice
c. SyntaxError: missing slice index
d. TypeError: can't multiply sequence by int
e. none of the above

Question 4

```
noise = 'hullaballoo'
changeCount = 0
for i in range(1, len(noise)):
    if noise[i] != noise[i-1]:
        changeCount += 1
print(changeCount)
```

a. IndexError: string index out of range
b. 3
c. 4
d. 7
e. none of the above

Question 5

```
braveOnes = ['Merida', 'Elinor', 'Macintosh', 'Fergus']
vowels = 'aeiou'
for character in braveOnes:
    for vowel in vowels:
        if vowel in character:
            continue
        else:
            print(character[0], end='')
            break
```

a. no output
b. MMEEEMMFFF
c. MMMEEEMMMF
d. MEMF
e. none of the above

Question 6

```
pandemics = [['HIV'], '', ['Ebola', 'SARS'], ['polio', 'smallpox']]
print(pandemics[1:3])
```

a. 'IV'
b. ['HIV'], ''
c. '', ['Ebola', 'SARS']
d. '', ['Ebola', 'SARS'], ['polio', 'smallpox']
e. none of the above

Question 7

```
def dictTest(thing, aDict):
    if thing in aDict:
        return thing
    else:
        for v in aDict.values():
            if thing in v:
                return v

types = {'hero':['Alice', 'superman'],'sidekick':['rabbit', 'Jimmy']}
print(dictTest('Alice', types))
```

a. None
b. 'hero'
c. 'Alice'
d. ['Alice', 'superman']
e. none of the above

Question 8

```
boolExprs = [True and False, not True, True or False, True and not False]
fCount = 0
for expr in boolExprs:
    if expr == False:
        fCount += 1
    else:
        break
print(fCount)
```

a. SyntaxError: and not
b. 1
c. 2
d. 3
e. none of the above

Question 9

```
abe = 'no man has a good enough memory to be a successful liar'
def wordLens(t, limit):
    rtn = []
    aList = t.split()
    for s in aList:
        if len(s) > limit:
            return rtn
        rtn.append(len(s))
    return rtn

print(wordLens(abe, 5))
```

a. None
b. []
c. ['no', 'man', 'has', 'a', 'good']
d. 5
e. none of the above

Question 10

```
def fileCount(fileName, s):
    inF = open(fileName)
    count = 0
    for line in inF:
        count += line.lower().count(s)
    inF.close()
    return count

w = open('seuss.txt', 'w')
w.write('You have brains in your head' + '\n')
w.write('You have feed in your shoes' + '\n')
w.close()
print(fileCount('seuss.txt', 'you'))
```

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

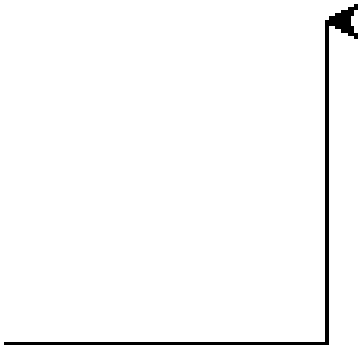
Programming questions 11-13 are worth 20 points each
Question 11a (8 points)

Write a function named *halfSquare* that uses turtle graphics to draw half of a square on the screen. This involves drawing two lines of equal length, making a 90 degree turn after each. For full credit, these repeated operations should be performed in a loop.

The function *halfSquare* takes two parameters

i. *t*, a turtle that is used for drawing
ii. *length*, the length of the lines

The function *halfSquare* should begin drawing without changing the position or orientation of the turtle *t*. The turtle *t* that is passed to *halfSquare* may initially be either up or down and may be at any location on the screen and in any orientation.



The figure at the right shows sample graphical output of *halfSquare*.

Question 11b (12 points)

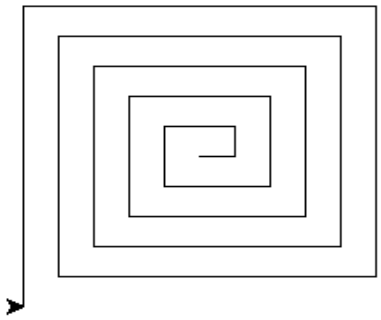
Write a function named *halfSquares* that calls the function *halfSquare* repeatedly to draw connected half squares of increasing size. (You can use this technique to draw a spiral pattern.)

The function *halfSquares* takes four parameters:

i. *t*, a turtle used for drawing
ii. *initial*, the length of a side of the first half square
iii. *increment*, the increase in side length of each successive half square
iv. *reps*, the number of half squares to draw

For example, the graphical output of the code below should look like the illustration at the right:

```
import turtle
s = turtle.Screen()
turt = turtle.Turtle()
halfSquares(turt, 20, 20, 10)
```



Question 12 (20 points)

Write a function named *wordCount* that counts how many words there are in each line of an input file and writes that count to a corresponding line of an output file. The input file already exists when *wordCount* is called. *wordCount* creates the output file.

Input. The function *wordCount* takes two parameters:

i. *inFile*, a string that is the name of a text file that is to be read and analyzed. The file that *inFile* refers to contains only upper and lower case letters and white space (no punctuation marks or other special characters).
ii. *outFile*, a string that is the name of the file to which *wordCount* writes its output.

The input file is in the current working directory and you should create the output file to that directory.

Output. For each line of *inFile*, *wordCount* should write a corresponding line to *outFile* containing a single integer: the number of words on the line.

If the content of the file catInTheHat.txt is below,

The sun did not shine
It was too wet to play
So we sat in the house
All that cold cold wet day
I sat there with Sally
We sat there we two
And I said How I wish
We had something to do

the function call

```
wordCount('catInTheHat.txt', 'catInTheHatOut.txt')
```

should create a file named catInTheHatOut.txt with this content:

5
6
6
6
5
5
6
5

Question 13 (20 points)

Write a function named *initialVowels* that analyzes the contents of a text file for words that begin with a vowel. The letters a, e, i, o and u are vowels.

The function *initialVowels* should return a dictionary of vowel:word-list pairs. Each vowel should be a key in the dictionary if and only if it is the first letter of some word in the input file. The value of a key is a list of all the words in the input file in which that vowel is the first letter. (Hint: your job is easier if you lowercase the text in the input file.)

Input. The function *initialVowels* takes a single parameter:

i. *inFile*, a string that is the name of a text file. This file contains only upper and lower case letters and white space (no punctuation marks or other special characters). It is in the current working directory.

Output. Return a vowel:word-list dictionary

For example, if the file named catInTheHat.txt contains the same text as in Question 12, then the function call

```
print(initialVowels('catInTheHat.txt'))
```

should output

```
{'i': ['it', 'in', 'i', 'i', 'i'], 'a': ['all', 'and']}
```