Question 1

```
for i in range(-3, 0):
    print(i, end=' ')
```

a. no output
b. -3 -2 -1 0
c. 0 -1 -2 -3
d. -3 -2 -1
e. none of the above

Question 2

```
mess = 'hodge podge'
count = 0
idx = 0
while idx < len(mess):
    if mess.count(mess[idx]) > 1:
        count += mess.count(mess[idx])
    else:
        count += 1
    idx += 1
print(count)
```

a. 8
b. 9
c. 11
d. 18
e. none of the above

Question 3

```
s = 'sliceMagic'
print(s[:] + s['M':] + s[:'M'])
```

a. TypeError: slice indices must be integers
b. sliceMagicMagicslice
c. sliceMagicMagicsliceM
d. Magicslice
e. none of the above

Question 4

```
parting = 'too-da-loo'
aList = ['oo', '-', 'oo']
inCount = 0
for i in range(len(parting)-1):
    if parting[i] + parting[i+1] in aList:
        inCount += 1

print(inCount)
```

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

Question 5

```
xChars = ['Mystique', 'Wolverine']
vowels = 'aeiou'
for vowel in vowels:
    for char in xChars:
        if vowel not in char:
            continue
        print(char[0], end='')
```

a. the empty string
b. MW
c. MWMMW
d. MWMWWM
e. none of the above

Question 6

```
pandemics = [['HIV', 'Ebola', 'SARS'], ['polio', 'smallpox'], 'flu']
print(pandemics[2:])
```

a. SARS
b. ['SARS']
c. ['smallpox']
d. flu
e. none of the above

Question 7

```
def returnDictVal(d, aKey):
    if aKey in d:
        return d[aKey]
    else:
        return None

chars = {'Alice':['rabbit', 'caterpiller'], 'superman':'Clark'}
wonderWomanChars = {'wonder woman':['Lynda Carter', 'Gal Gadot']}
print(returnDictVal(chars, 'wonder woman'))
```

a. the empty string
b. None
c. NameError: name 'returnDictVal' is not defined
d. ['Lynda Carter', 'Gal Gadot']
e. none of the above

Question 8

```
bools = [True or False, True, True and False, True and not False]
trues = 0
for bool in bools:
    if bool == True:
        trues += 1
        continue
    break
print(trues)
```

a. TypeError: bool takes 1 positional argument but 2 were given
b. 0
c. 2
d. 3
e. none of the above

Question 9

```
anonymous = 'hollywood is the opium of the people'
def nonSubStr(t, subStr):
    rtn = []
    aList = t.split()
    for s in aList:
        if subStr not in s:
            rtn.append(s)
    return rtn

print(nonSubStr(anonymous, 'o'))
```

a. False
b. []
c. ['is']
d. ['is', 'the']
e. none of the above

Question 10

```
def inFileCount(fileName, searchStr):
    inF = open(fileName)
    contents = inF.read()
    inF.close()
    return contents.count(searchStr)

seuss = open('you.txt', 'w')
seuss.write('Today you are You, that is truer than true.' + '\n')
seuss.write('There is no one alive who is Youer than You.' + '\n')
seuss.close()
print(inFileCount('you.txt', 'You'))
seuss.close()
```

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

Question 11a
8 points

Write a function named rectangle() that uses turtle graphics to draw a rectangle of specified dimensions. The function rectangle() takes three parameters:

i. *t*, a turtle that is used for drawing
ii. *size1*, the length of the first (and third) side of the rectangle
iii. *size2*, the length of the second (and fourth) side of the rectangle

The function rectangle() should draw a rectangle beginning at the initial position and orientation of *t*, and should leave *t* with the same position and orientation on exit. Do not make any assumptions about the initial state of the turtle. For full credit you must use a loop for repeated operations.

Question 11b
12 points

Write a function named panels() that uses turtle graphics and the function rectangle() (Question 11a) to draw a sequence of rectangles of specified size and orientation. Begin drawing each rectangle at the initial location of the passed turtle parameter. Rotate each rectangle after the first one counterclockwise by a specified angle. Make the second (and fourth) side of each rectangle twice the length of the first (and third) side.

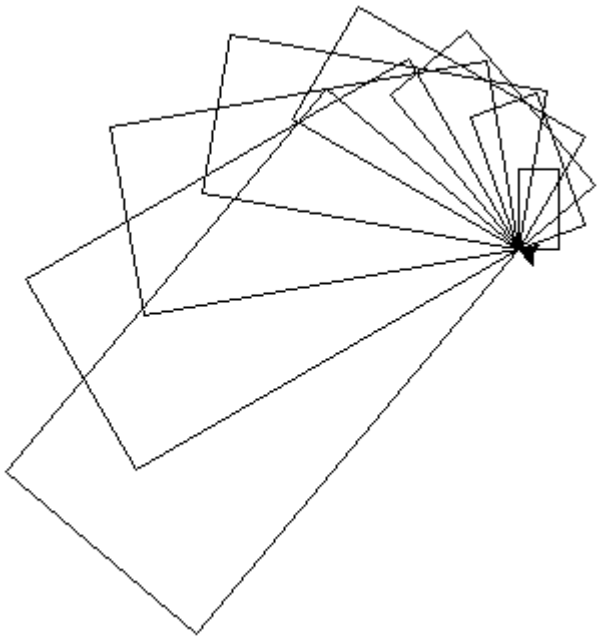
The function panels() should repeatedly call rectangle() to draw rectangles of increasing size, rotated by a specified angle.

The function panels() takes 5 parameters:

i. *t*, a turtle used for drawing
ii. *initSize*, the length of the short side of the first rectangle
iii. *delta*, the increase in length of the short side of successive rectangles
iv. *numPanels*, the number of rectangles to draw
v. *angle*, the number of degrees to rotate successive rectangles

If panels() is called by the following code, the graphic below would be correct output. (Hint: panels() should draw with the passed turtle, whatever location and orientation the turtle is in.)

```
import turtle
s = turtle.Screen()
shelly = turtle.Turtle()
panels(shelly, 20, 15, 8, 20)
```



Question 12

The letters a, e, i, o and u are the only vowels. Write a function named vowelUseDict() takes a string *t* as a parameter and computes and returns a dictionary with the number of words in *t* containing each vowel. Assume that the given text contains only lower case letters and white space.

Input: *t*, a string consisting of lower case letters and white space
Return: a dictionary in which each vowel is a key and its value is the number of words containing that vowel

For example, the following would be correct output.

```
text = 'like a vision she dances across the porch as the radio plays'
print(vowelUseDict(text))
{'e': 5, 'u': 0, 'o': 4, 'a': 6, 'i': 3}
```

Question 13

Write a function named longestWord() that find the length of the longest word in each line in a file and writes that length to a new file.

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

Assume that the input file contains only letters and white space. Assume that the input file is in the current working directory and write the output file to that directory.

The function longestWord() should write a line to the output file only if the line in the input file is not empty, that is, it contains at least one word. For example, if the following is the content of the file trouble.txt:

We surely got trouble
Right here in River City

Gotta figger out a way
To keep the young ones moral after school

The following function call:

```
inF = 'trouble.txt'
outF = 'troubleLongest.txt'
longestWord(inF, outF)
```

should create the file longestWord.txt with the content:

7
5
6
6