CS 100 2019F
Midterm 1: Practice Exam 1

Multiple choice questions. 4 points each.

Question 1

```
time_machine = False
system_restore = False
recover = True

if not time_machine and not system_restore:
    print('unknown configuration!')
elif not time_machine:
    print('windows')
else:
    print('mac')
if time_machine and system_restore:
    print('stranger than fiction!')
elif recover:
    print('reboot the system')
```

a) `unknown configuration!`

b) `unknown configuration!`
`stranger than fiction!`

c) `windows`
`reboot the system`

d) `mac`
`reboot the system`

e) none of the above

Question 2

```
letters = ['s', 'r', 'a', 'c', 's', 'o']
concat = letters[-5] + letters[2] + letters[-3]
print(concat)
```

a) `rac`

b) `rrc`

c) `sra`

d) `IndexError: string index out of range`

e) none of the above

Question 3

```
fox = ['the', 'truth', 'is', 'out', 'there']
print(fox[4:5])
```

a) `['there']`

b) `'there'`

c) `there`

d) `'out'`

e) none of the above

Question 4

```
chain = [1, '2', 3, '4']
leading = chain[:1]
trailing = chain[-1:]
print(leading + trailing)
```

a) `5`
b) `14`
c) `23`
d) `TypeError: unsupported operand type(s) for +: 'int' and 'str'`
e) none of the above

Question 5

```
import turtle
s = turtle.Screen()
t = turtle.Turtle()
for i in range(7):
    if i%4 == 0:
        t.forward(50)
        t.right(90)
```

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

Question 6

```
def validate(password, ideal):
    if len(password) < ideal - 3:
        return 'too short'
    elif len(password) > ideal + 3:
        return 'not optimal'
    else:
        return 'perfect!'
    return 'flawed'

print(validate('icanttellyou', 8))
```

a) `too short`
b) `not optimal`
c) `perfect!`
d) `flawed`
e) none of the above

Question 7

```
def aggregate(sequence):
    result = []
    for element in sequence:
        result.append(element)
    return result

print(aggregate('bubbles'))
```

a) ['bules']
b) ['bubbles']
c) ['b', 'u', 'l', 'e', 's']
d) ['b', 'u', 'b', 'b', 'l', 'e', 's']
e) none of the above

Question 8

```
repeats = 0
phrase = 'Oh hot diggity'
word = 'dog'
for letter in phrase:
    if letter in word:
        repeats += 1

print(repeats)
```

a) 2
b) 3
c) 4
d) 5
e) none of the above

Question 9

```
def rangeExploration(sequence, begin, end, step):
    result = ''
    for i in range(begin, end, step):
        result += sequence[i]
    return result

s = 'oops aardvarks eat ants'
print(rangeExploration(s, 1, 9, 2))
```

a) osav
b) osarv
c) osar
d) op ad
e) none of the above

Question 10

```
def sliceAndDice(name):
    mystery = []
    recycle = []
    for item in name:
        if item in mystery:
            recycle.append(item)
        else:
            mystery.append(item)
    return len(mystery)

print(sliceAndDice(['t', 'e', 'nn', 'e', 'ss', 'ee']))
```

a) 1
b) 4
c) 8
d) 9
e) none of the above

Question 11A (10 points)

A square wave is a wave that transitions between two extremes using slanted lines.

Write the function *squareWave* that draws a square wave using 5 line segments of equal length.

The function *squareWave* takes two parameters:

1. *t*, a turtle that is used to draw the square wave. The turtle *t* may be in any position, orientation and up/down state.
2. *length*, a positive integer that is the length of one line segment

The function *squareWave* should:

1. draw a straight line, a fall, a straight line, a rise, and a straight line, in that order, with a 45 degree acute angle between the straight lines and the slanted lines
2. leave *t* positioned at the end of the last straight line when it returns
3. leave *t* in its initial orientation when it returns

(Hint: this position and orientation leave the turtle ready to draw another square wave.)

For example, the following would be correct graphical output.

```
import turtle
snappy = turtle.Turtle()
squareWave(snappy, 50)
```

Question 11B (10 points)

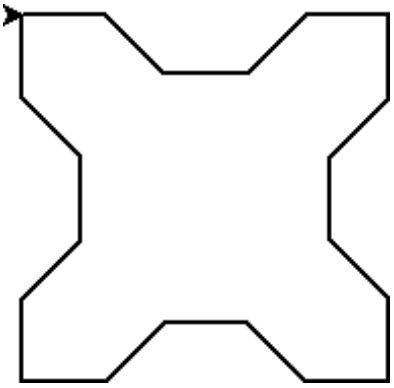
Write a function named *motif* that calls *squareWave* to draw a specified number of square waves.

The function *motif* takes four parameters:

- a turtle, *t*
- an integer *length*, that is the length of one line segment
- an integer *num*, the number of square waves to draw
- an integer *angle*, the clockwise rotation between successive square waves

For example, the following would be correct output.

```
import turtle
snappy = turtle.Turtle()
motif(snappy, 50, 4, 90)
```



Question 12 (20 points)

A word is divergent if it begins and ends with the same letter of the alphabet. For example, the word `"area"` is divergent, but the word `"land"` is not.

Write a function named *`getDivergent`* that computes and returns how many words in a list of words are divergent and how many are not.

Input: The function *`getDivergent`* takes a single parameter, *`words`*, that is a list of non-empty, lowercase strings.

Output: The function *`getDivergent`* should return a pair of integers in the form of a tuple, the first being the number of divergent words in *`words`* and the second being the number of non-divergent words in *`words`*.

For example, the following would be correct output.

```
>>> advice = ['the', 'secret', 'to', 'a', 'better', 'health', 'is', 'exercise']
>>> print(getDivergent(advice))
(3, 5)
```

Question 13 (20 points)

Write a function named *`printLines`*. The function *`printLines`* should:

- takes one parameter: a string named *`line`*
- prompt the user for an item of input: the number of times to print *`line`*
- produce two kinds of output. It should:
    1. print out *`line`* the desired number of times
    2. return the total number of lines it printed

For example, the following would be correct input and output:

```
>>> count = printLines('#####')
Number of repetitions? 3
#####
#####
#####
>>> print(count)
3
```