CS 100 2019F
Midterm 02: Practice Exam 1

Questions 1-10 are multiple choice. 4 points each.

Question 1

```
def compare(s, wordList):
    rtnList = []
    for word in s.split():
        if word in wordList:
            continue
        if word[0] == word[-1]:
            break
        rtnList.append(word)
    return len(rtnList)

bill = "big dreams create the magic that stir souls to greatness"
print(compare(bill, ['to', 'the', 'that']))
```

a) `4`
b) `5`
c) `6`
d) `7`
e) none of the above

Question 2

```
for i in range(1, 3):
    for j in range(1, 2):
        print('#', end=' ')
```

a) `#`
b) `# #`
c) `# # #`
d) `# # # #`
e) none of the above

Question 3

```
boolList = [True or False, not False, not not True, 9//2 == 4.5, False, True]
i = 0
while boolList[i]:
    i += 1
print(i)
```

a) `IndexError: list index out of range`
b) `2`
c) `3`
d) `4`
e) none of the above

Question 4

```
def asymmetry(s):
    symmetric = ''
    for i in range(len(s)):
        if s[i] == s[-i-1]:
            symmetric += s[i]
        else:
            return s[i:-i]
    return symmetric

t = 'sambas'
print(asymmetry(t))
```

a) `mb`
b) `mbsaas`
c) `saas`
d) `sambas`
e) none of the above

Question 5

```
menu = [{1:['Salad', 'Soup']}, {2:['Entree']}, {3:{'Desert':257, 'Fruit':95}}]
print(menu[1][2][3])
```

a) `KeyError: 1`
b) `KeyError: 2`
c) `KeyError: 3`
d) `u`
e) none of the above

Question 6

```
activities = ['eating', 'sleeping']
vowels = 'aeiou'
for vowel in vowels:
    for activity in activities:
        if vowel in activity:
            print(activity + ' ', end='')
        else:
            break
```

a) `eating sleeping`
b) `eating sleeping eating sleeping`
c) `eating sleeping eating sleeping eating sleeping`
d) `eating eating sleeping eating sleeping`
e) none of the above

Question 7

```
outF = open('digits.txt', 'w')
for i in range(2):
    outF.write(str(i))
outF.close()

inF = open('digits.txt', 'r')
for line in inF:
    print(line)
inF.close()
```

a) `01`

b)
```
0
1
```

c) `012`

d)
```
0
1
2
```

e) none of the above

Question 8

```
words = ['the', 'dust', 'which', 'the', 'wind', 'blows', 'in', 'your', 'face']

extremelyOdd = ''
for word in words:
    length = len(word)
    if length % 2 == 1 or length > len(extremelyOdd):
        extremelyOdd = word

print(extremelyOdd)
```

a) `the`

b) `dust`

c) `which`

d) `blows`

e) none of the above

Question 9

```
import turtle

t = turtle.Turtle()
for i in range(4):
    if i%2 == 0:
        t.up()
        t.forward(100)
        t.right(90)
    if i%3 == 0:
        t.down()
        t.forward(100)
        t.right(90)
```

a) a straight line

b) two parallel lines

c) two adjacent sides of a square

d) three sides of a square

e) none of the above

Question 10

```
def inFileCount(fileName):
    inF = open(fileName)
    length = len(inF.read())
    inF.close()
    return length

outF = open('inspire.txt', 'w')
outF.write("Don't be the same." + "\n")
outF.write("Be better." + "\n")
outF.close()
print(inFileCount('inspire.txt'))
```

a) 24

b) 28

c) 30

d) 32

e) none of the above

Question 11A (8 points)

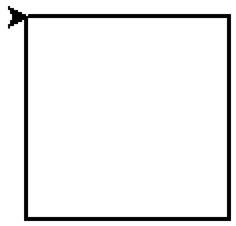
Write a function named *drawSquare*. The function *drawSquare* takes two parameters:

1. a turtle, *t*
2. an integer, *Length*, that is the length of a side of the square

The function *drawSquare* should use the turtle *t* to draw the square. It should begin drawing with the turtle at its initial position and orientation. After drawing a side, the turtle should turn clockwise. When *drawSquare* exits, the turtle should again be in its initial position and orientation. Do not make any assumptions about the initial up/down state of the turtle, its position on the screen or its orientation.

You must use a loop for repeated operations.

```
import turtle
t = turtle.Turtle()
drawSquare(t, 100)
```



Question 11B (12 points)

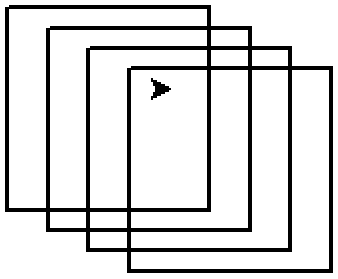
Write a function named *offsetSquares* that uses turtle graphics and the function *drawSquare* in question 11A to draw a sequence of squares.

The function *offsetSquares* takes five parameters:

1. a turtle, *t*
2. an integer, *Length*, that is the length of a side of a square
3. an integer *num*, the number of squares to draw
4. an integer *xOffset*, the offset parallel to the initial orientation relative to the previous square
5. an integer *yOffset*, the offset perpendicular to the initial orientation relative to the previous square

If *offsetSquares* is called by the following code, this would be correct output:

```
import turtle
t = turtle.Turtle()
offsetSquares(t, 100, 4, 20, 10)
```



Question 12 (20 points)

Write a function named *indexByLength* that takes one parameter:

- *s*, a string

The function *indexByLength* returns a dictionary in which each key is the length of a word in *s* and the corresponding value is a list of all distinct words in *s* of that length. Even if a word occurs more than once in *s*, it should appear in a value list only once. Treat the text as case sensitive (e.g., "Python" and "python" are two different words). Assume, for simplicity, that *s* contains only letters, and spaces.

The following is correct sample output:

```
>>> text = 'I was indecisive but now I am not too sure'
>>> print(indexByLength(text))
{1:['I'], 3:['was', 'but', 'now', 'not', 'too'], 10:['indecisive'], 2:['am'], 4:['sure']}
```

Question 13 (20 points)

Write a function named *cloneLines* that takes two parameters:

1. *inFile*, a string, the name of an input file that exists before *cloneLines* is called
2. *outFile*, a string, the name of an output file that *cloneLines* creates and writes to

The function *cloneLines* reads the content of *inFile*, line by line, and writes into *outFile* any line that contains at least one word that occurs more than once on the line. You may assume that the input file contains only lowercase letters, spaces and newline characters.

For example, if the following is the content of the file *william.txt*:

```
double double toil and trouble
fire burn and caldron bubble

eye of newt and toe of frog
fillet of a fenny snake
in the caldron boil and bake

double double toil and trouble
fire burn and caldron bubble
```

The following function call:

```
>>> inFile = 'william.txt'
>>> outFile = 'clones.txt'
>>> cloneLines(inFile, outFile)
```

should create the file *clones.txt* with the content:

```
double double toil and trouble
eye of newt and toe of frog
double double toil and trouble
```