Python - MOCK EXAM

Introduction to Coding Using Python
Mock Exam
Time allowed: 60 minutes
Note: Main Exam will have 5 question and 90 minutes

Marking criteria

Each question will be marked according to the following criteria:

· *Does it work? This will be checked by running a series of test cases. All tests must pass to receive the marks. No partial marks will be given. You will be given a full set of test cases so that you can check your code before submitting it. Tests passed by use of hard coding will be deemed to have failed.*

· *Is it written using clean code?*

· ***You must provide your answer in this form and save each in your local PythonWorkspace for review by your trainer.***

Clean code

· *Naming standards:*
o All variables start with a lower case letter and use camel case.
o Clear descriptive variable names with no ambiguity. E.g. a float containing a price should be called 'price' but a list containing some prices should be called 'prices'
o The function name is identical to the name specified in the question. If it isn't it will fail the tests.

· *No obsolete lines of code:*
o So no print statements in the function
o No code which has no impact on the return value

· ***No input statements** in the function*
· ***No print statements** in the function*
· *No commented out lines of code*
· *No excessively complex code. Think about breaking code down into smaller, simpler functions if it gets too complex.*

Points: -/100

1. Q1 – Right Angled Triangle

write a function called rightAngledTriangle which accepts the lengths of three sides of a triangle as parameters. The function should return whether or not the triangle is a **right angled triangle**.

Arguments:

- An integer (the length of side 1)
- An integer (the length of side 2)
- An integer (the length of side 3)

Return value:

- True if the sides make an right angled triangle.
- False if the sides do NOT make an right angled triangle.
- False if any side is 0 or negative.

* (25 Points)

```
def rightAngleTriangle(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        return False
    elif a ** 2 + b ** 2 == c ** 2 or c ** 2 + b ** 2 == a ** 2 or a ** 2 + c ** 2 == b ** 2:
        return True
    else:
        return False
```

2. Q2 – Calculate Factorial *

write a function called calculateFactorial which will return the factorial of a given integer. The factorial of an integer N is the product of the integers between 1 and N, inclusive.

Arguments:

- An integer (N)

Return value:

- An integer (the factorial of the number)

For example:

- If N = 5 then N factorial is 120 which is: 1 * 2 * 3 * 4 * 5

Note:

- The factorial of 0 (zero) is: 1

- The factorial of any negative number is invalid – so return 0 (zero)

(25 Points)

```
def calculateFactorial(N):

    startNumber = 1
    multiply = 1

    while startNumber < N:
        startNumber += 1
        multiply = startNumber * multiply
    return multiply
```

3. Q3 – Count words that end in R or S

write a function called countWords which takes as the input a string of words and returns a count of the words ending in r or s
So the r in paper and the s in files count, but not the r in red

Note: Change the input string to lowercase so that R and r is the same

Arguments:
- A string of words (not case sensitive with spaces between them)

Return value:
- An integer (containing the count of words ending in r or s)

For example:
- "paper chase" would return 1 (paper ends with r)
- "red paper files" would return 2 (paper and files)
- "sss rrrrr abcdef" would return 2 (sss and rrrrr)

* (25 Points)

```
wordString1 = 'paper and the s in files count'
wordString2 = 'paper paper and the s in files count'

def countWords(wordString):
    x= wordString.split()
    count = 0
    for entry in x:
        if entry[-1] == 'r':
            count += 1
    return count
```

4. Q4 – Find duplicates in a string

write a function called findDuplicates which accepts a string and returns the count of how many duplicates there are in the string.

Arguments:

- A string (of any length containing letters, numbers and symbols)

Return value:

- An integer (containing the count of **unique** duplicates in the string)

For example:

bccbbbbbb	would return 2 (b and c are duplicated)
abcdef	would return 0 (there are no duplicates)
HGF hdgf	would return 0 (there are no duplicates)
##12ab	would return 1 (the # is duplicated)

Note: Treat the string as case sensitive. So ABC is not the same as abc
* (25 Points)

```
def findDuplicates(aString):
    for x in set(aString):
        if aString.count(x) > 1:
            print('count: ', aString.count(x))
            print('x:', x)
```