PSET 3: Manufacturing Data String Decoding Exercise

Quality Control (QC) Testing String Decoding

For this assignment, suppose an automated manufacturing tracker records production QC test results in batches in a results string. For example,

Q2p1d1

would indicate that a single batch of two QC tests was performed with the results of one pass and one defect. Note that the character Q is used to start a batch of QC test results. The character p is to identify the number of tests that passed the QC test, and the character d indicates the number of defects. More than one batch of QC test results can be reported in a single results string. For example,

Q2d1p1Q5p3d2

would indicate two batches of QC test results, one with two test results and one with five test results, with a combined total of four passes and three defects.

Precisely, to be a valid QC test results string,

- a batch of results must begin with the character Q (case sensitive)
- a batch of results must report both pass and defect test results with p (case sensitive – lower case) and d (case sensitive – lower case) in either order
- no leading zeros are allowed in any numeric value being reported
- the total number of QC tests in a batch must equal the number of pass and defect test results.
- the total number of QC tests in a batch must be greater than zero (0).
- a single result string may include multiple batches of results

All of the following are examples of valid result strings:

- Q1p0d1Q1d0p1 (two batches of results, two total tests, one pass, one defect)
- Q5d2p3 (one batch of results, five total tests, two defects, three passes)

All of the following are examples of invalid result strings:

- q1p0d1 (batch must be reported with Q)
- Q1pd1 (a number for pass is required)
- Q1p1d (a number for defect is required)
- Q1p0d1 asdfR (extra characters not allowed)
- Q5p00003d0002 (leading zeros not allowed)
- Q5p0d0 (pass and defect results must equal the total number of tests)
- Q0p0d0 (batch cannot be zero)

Your task

For this project, you will implement the following function, using the exact function name, parameter type, and return type shown in this specification. (The parameter and variable *names* may be different if you wish. Again the function name must be isValidString).

```
def isValidString(s):
    return isValid
```

s is the QC test results string (see above examples of valid and invalid test result strings.)

isValid is a boolean variable that stores either a True or False value.

This function returns true if its parameter s is a well-formed test result string described above, or false otherwise.

You should include an adequate level of comments in your code to help the code reviewer understand your code.

Save your Python program as QCTestString.py. Your QCTestString.py should include the isValidString() function and other supporting functions that you developed and called from the isValidString() function. Please remove all output statements (print()) from your functions if you used any output statements like print() to help you to test and debug.

IMPORTANT: You can only use Python built-in functions to write this program. You are not allowed to use any imported modules, packages, or libraries such as pandas, NumPy, math, etc. the goal of this assignment is to practice (You may or may not use all of these listed Python features):

- Program design
- User-defined functions with a return value
- Global variables
- Boolean variables
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- If-elif-else control statements including nested ifs.
- While loops
- Data type casting
- String index, built-in string len() and isdigit() or split() functions

You should be able to complete the assignment with the above-mentioned built-in Python features.

Make sure that you test your function with both valid and invalid test result strings. We will be testing your function with 30 test cases (QC test result strings).

The grading rubric is as follows:

- The assignment is worth 100 points. Grading is based on function execution and generating correct results. You will get zero points if the function does not execute or fails all 30 test cases.

You'll get a detailed solution from a subject matter expert that helps you learn core concepts.

For this assignment, suppose an automated manufacturing tracker records production QC test results in batches in a results string. For example,
Q2p1d1

would indicate that a single batch of two QC tests was performed with the results of one pass and one defect. Note that the character Q is used to start a batch of QC test results. The character p is to identify the number of passes and the character d is to identify the number of defects. More than one batch of QC test results can be reported in a single results string. For example,
Q2d1p1Q5p3d2

- Each test case is worth 3.3333 points (for a max score of 100).
- Your adequate comments and code readability are expected. No additional points will be given for adequate comments and code readability. However, we will review your code and subtract up to 6 points from the overall assignment grade for the lack of adequate comments or code readability.

Hi, class,

I just finished the grading of PSET 3. Most tests I created are easy. I am glad most of you did a great job, even when some of you have never used python before.

Please don't be discouraged! We always want you enjoy learning programming and gain some Python experience. For those get <80%, please keep working on this PSET 3. You can directly send me the revised version to get extra credits.

Also, we will have an optional project that you can submit for extra credit.

Below are the tests I created. First 20 tests should return False.

```
print ('1', isValidString('XYZ'))
print ('2', isValidString('Qpd'))
print ('3', isValidString('QQQ'))
print ('4', isValidString('ppp'))
print ('5', isValidString('ddd'))
print ('6', isValidString('1234567890'))
print ('7', isValidString('q20P18D2'))
print ('8', isValidString('Q20P18D2'))
print ('9', isValidString('Q8 p6d2'))
print ('10', isValidString('!@#$%^&*'))
print ('11', isValidString('Q9p5d5'))
print ('12', isValidString('Q9d5p5'))
print ('13', isValidString('Q0p0d0'))
print ('14', isValidString('Q5Q5d4'))
print ('15', isValidString('Q22p20d2QQ4p2d2'))
print ('16', isValidString('Q22p20d2Q40p2d2'))
print ('17', isValidString('Q22p20d2Q4p2d2 '))
print ('18', isValidString('Q9p5d4Q4p1d2'))
print ('19', isValidString('Q22p20d2pQ4p2d2'))
print ('20', isValidString('Q5p2d3Q5p2p3'))
print ('21', isValidString('Q5p2d3Q5p2d3Q5p2d3Q5p2d3Q5p2d3'))
print ('22', isValidString('Q5p2d3Q5p2d3'))
print ('23', isValidString('Q18p14d4'))
print ('24', isValidString('Q18p14d4Q18p14d4'))
print ('25', isValidString('Q8p4d4Q8p4d4'))
print ('26', isValidString('Q1p1d0'))
print ('27', isValidString('Q1p0d1'))
print ('28', isValidString('Q8p4d4'))
print ('29', isValidString('Q3p3d0'))
print ('30', isValidString('Q9p5d4'))
```