Question 1. [7 marks]

The following python statements have been executed:

tweet1 = 'Computer science students organize #UofT Hacks to create new solutions'
tweet2 = 'Meet Ross, the @IBMWatson-powered lawyer @IBM'

# docstrings removed to save space
def f(x, y):
    return x + y + 1

def g(x, y):
    return x // y

x = 10
y = 8

Beside each code fragment in the table below, write what is printed when the code fragment is executed. If the code would cause an error, write ERROR and give a brief explanation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Output or Cause of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>print('uoft' in tweet1 or 'Ross' not in tweet2)</td>
<td>False</td>
</tr>
<tr>
<td>print(10 // 3 * 3 + 10 % 3 == 10)</td>
<td>True</td>
</tr>
<tr>
<td>print(tweet2[-3:])</td>
<td>IBM</td>
</tr>
<tr>
<td>print(tweet2[0].isupper() or tweet2[len(tweet1)].isupper())</td>
<td>True</td>
</tr>
<tr>
<td>print(tweet2[len(tweet1)].isupper() or tweet2[0].isupper())</td>
<td>ERROR: index out of range</td>
</tr>
<tr>
<td>print(g(x, x))</td>
<td>1</td>
</tr>
<tr>
<td>print(f(y, g(x, x)))</td>
<td>10</td>
</tr>
</tbody>
</table>
**Question 2.** [3 marks]

Assume that variable `s` refers to a `str`. Consider the following Python code:

```python
VOWELS = 'aeiou'
v_count = 0
for ch in s:
    if ch in VOWELS:
        v_count = v_count + 1
```

Rewrite this code so that it uses a `while` loop instead of a `for` loop.

```python
VOWELS = 'aeiou'
v_count = 0
i = 0
while i < len(s):
    if s[i] in VOWELS:
        v_count = v_count + 1
    i = i + 1
```
Question 3.  [8 marks]

The City of Toronto sets the fares that taxi companies are allowed to charge for rides within the city. The following amounts are charged:

- For the first 0.143 km or part thereof: $4.25
- For each additional 0.143 km or part thereof: $0.25
- For waiting time while under engagement, for each 29 seconds: $0.25
- For each additional passenger in excess of four: $2.00

For a group of three travelling a distance of 0.286 km, with a waiting time of 15 seconds, the charges would be: $4.25 for the first 0.143 km, $0.25 for the next 0.143km, and $0.25 for the waiting time, giving a total of $4.75. If the distance had been 0.001 km longer, the cost would go up by $0.25, since the trip would be into an additional part of 0.143 km.

Using the Function Design Recipe that you have learned in the course, write a complete python function named `compute_taxi_fare` that returns the fare that would be charged for a given number of passengers to travel a given distance in km with a given waiting time in seconds. Use a python expression of the form `round(x,2)` to round a fare to two digits after the decimal point.

Your function may assume that when it is called, it is given valid arguments (postive number of passengers, positive distance travelled, and a non-negative waiting time). You may assume that the `math` module has been imported.

Use the following space for rough work. The code written on this page will not be marked unless you indicate that it should be.
def compute_taxi_fare(num_passengers, distance, wait_time):
    """ (int, number, number) -> float

    Return the taxi fare for num_passenger passengers to travel distance km
    with a waiting time of wait_time seconds.
    """
    if distance <= 0.143:
        fare = 4.25
    else:
        excess = distance - 0.143
        fare = 4.25 + math.ceil(excess / 0.143) * 0.25
        # alternate:
        # if excess % 0.143 == 0: # alternate: excess / 0.143 == excess // 0.143
        #     fare = 4.25 + excess // 0.143 * 0.25
        # else:
        #     fare = 4.25 + (excess // 0.143 + 1) * 0.25

    fare = fare + math.ceil(wait_time / 29) * 0.25
    # alternate:
    # if wait_time > 0: # check may not be required but is if // used
    #     if wait_time % 29 == 0: # alternate: wait_time / 29 == wait_time // 29
    #         fare = fare + wait_time // 29 * 0.25
    #     else:
    #         fare = fare + (wait_time // 29 + 1) * 0.25

    if num_passengers > 4:
        fare = fare + (num_passengers - 4) * 2.00

    return round(fare,2)
Question 4. [7 marks]

When a character is repeated consecutively 0 or more times in a string, we call the sequence of repeated characters a run. For example, 'abbbcaacbb' contains 6 runs: 'a', 'bbb', 'cc', 'aa', 'c' and 'bb'. The length of the longest run in 'abbbcaacbb' is 3.

Complete the function according to its docstring description.

def longest_run(s):
    """ (str) -> int

    Return the length of the longest run of characters in s.
    """

    if len(s) == 0:
        return 0

    runs_list = []
    current_run = 1
    for i in range(1, len(s)):
        if s[i] == s[i-1]:
            current_run = current_run + 1
        else:
            runs_list.append(current_run)
            current_run = 1

    # add last run to the list
    runs_list.append(current_run)
    return max(runs_list)

    # alternate that does not use a list
    # if len(s) == 0:
    #     return 0
    #
    # longest = 1
    # current_run = 1
    # for i in range(1, len(s)):
    #     if s[i] == s[i-1]:
    #         current_run = current_run + 1
    #     else:
    #         if current_run > longest:

>>> longest_run('abcccddefg')
3
>>> longest_run('abcde')
1
>>> longest_run('')
0

"""
# longest = current_run
# current_run = 1
#
# return max(longest, current_run)