This midterm consists of 4 questions on 8 pages (including this one). When you receive the signal to start, please make sure that your copy is complete:

- Comments are not required except where indicated, although they may help us mark your answers.

- No error checking is required: assume all user input and all argument values are valid.

- If you use any space for rough work, indicate clearly what you want marked.

- You may use a pencil; however, work written in pencil will not be considered for remarking.

# 1: _____/ 7
# 2: _____/ 3
# 3: _____/ 8
# 4: _____/ 7

TOTAL: _____/25
Question 1.  [7 marks]

The following python statements have been executed:

tweet1 = 'Computer science students organize #UofTHacks 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></td>
</tr>
<tr>
<td>print(10 // 3 * 3 + 10 % 3 == 10)</td>
<td></td>
</tr>
<tr>
<td>print(tweet2[-3:])</td>
<td></td>
</tr>
<tr>
<td>print(tweet2[0].isupper() or tweet2[len(tweet1)].isupper())</td>
<td></td>
</tr>
<tr>
<td>print(tweet2[len(tweet1)].isupper() or tweet2[0].isupper())</td>
<td></td>
</tr>
<tr>
<td>print(g(x, x))</td>
<td></td>
</tr>
<tr>
<td>print(f(y, g(x,x)))</td>
<td></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.

You will not need to use much of the blank space on this page.
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 (positive 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.
(Write your solution to Question 3 on this page.)
[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]
Question 4. [7 marks]

When a character is repeated consecutively 0 or more times in a str, 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.

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

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

    >>> longest_run('abbbcaacbb')
    3
    >>> longest_run('abcde')
    1
    >>> longest_run('')
    0
```

Short Python function/method descriptions:

```
__builtins__:
  int(x) -> int
      Convert x to an integer, if possible. A floating point argument will be truncated towards zero.
  len(x) -> int
      Return the length of list, tuple, or string x.
print(value) -> NoneType
      Prints the values.
range(stop) -> list-like-object of int
range(start, stop[, step]) -> list-like-object of int
      Return the integers starting with start and ending with stop - 1 with step
      specifying the amount to increment (or decrement). If start is not specified,
      the sequence starts at 0. If step is not specified, the values are incremented by 1.
str(x) -> str
      Return an object converted to its string representation, if possible.
str:
  x in s -> bool
      Produce True if and only if x is in s.
S.count(sub[, start[, end]]) -> int
      Return the number of non-overlapping occurrences of substring sub in string S[start:end].
      Optional arguments start and end are interpreted as in slice notation.
S.find(sub[,i]) -> int
      Return the lowest index in S (starting at S[i], if i is given) where the
      string sub is found or -1 if sub does not occur in S.
S.isalpha() -> bool
      Return True if and only if all characters in S are alphabetic
      and there is at least one character in S.
S.isalnum() -> bool
      Return True if and only if all characters in S are alphanumeric
      and there is at least one character in S.
S.isdigit() -> bool
      Return True if and only if all characters in S are digits
      and there is at least one character in S.
S.islower() -> bool
      Return True if and only if all cased characters in S are lowercase
      and there is at least one cased character in S.
S.isupper() -> bool
      Return True if and only if all cased characters in S are uppercase
      and there is at least one cased character in S.
S.lower() -> str
      Return a copy of the string S converted to lowercase.
S.replace(old, new) -> str
      Return a copy of string S with all occurrences of the string old replaced with the string new.
S.startswith(sub) -> bool
      Return True if and only if S starts with the substring sub.
S.upper() -> str
      Return a copy of the string S converted to uppercase.
list:
  x in L -> bool
      Produce True if and only if x is in list L
L.append(object) -> NoneType
      Append object to end of list L.
```