Simon Fraser University
Computing Science 120
Midterm Examination – 2
CODING - SOLUTION
Summer 2019

Time: 45 minutes

Last/Family Name (please, print): ____________________________

First Name (please, print): ____________________________

Student Number: ____________________________

Signature: ____________________________

Instructor: Anne Lavergne

This examination has 6 pages.

Make sure you have a complete paper.

Instructions:

✓ No books, calculators, computers, cell phones, or other materials may be used.

✓ Read each question carefully before answering it.

✓ We must use Python 3.

✓ Let us always comment our code and use the Good Programming Style (GPS) discussed in class and posted on our course web site.

✓ We can assume that all Python code fragments given in this exam are syntactically correct, unless otherwise stated.

✓ The marks for each question are given in [ ].

Let us use this to manage our time:

• 1 mark correspond to 1 minute of work.

Good luck!
1. [10 marks] **Problem Statement:** In the space below, write a Python function called `numberOfElements(...)` that returns the number of elements found in its parameter (a tuple).
   
   Your function `numberOfElements(...)` must only take 1 parameter (a tuple) and it must not call the `len(...)` Python function. Make sure you examine everything that has been given to you below before you start writing your function.

   ```python
   # numberOfElements(...)  
   # Description: Function numberOfElements(...) returns the number of  
   # elements found in its parameter (a tuple).  
   # Author:         
   # Date: July 12, 2019
   
   Possible Solution:  
   https://repl.it/repls/HonoredGlassKeygen
   
   # Main part of the program - TestCase 1  
   tuple1 = (1,2,3,4,5,6,7,8,9)  
   print("The number of elements in this tuple {} is {}").format(tuple1, numberOfElements(tuple1))
   
   # TestCase 2  
   tuple2 = ("Brian", "Xiao", "Igor")  
   print("The number of elements in this tuple {} is {}").format(tuple2, numberOfElements(tuple2))
   
   # TestCase 3  
   tuple3 = ()  
   print("The number of elements in this tuple {} is {}").format(tuple3, numberOfElements(tuple3))
   
   Sample Runs (results of the 3 test cases in Main part of program):
   
   The number of elements in this tuple {1, 2, 3, 4, 5, 6, 7, 8, 9} is 9.  
   The number of elements in this tuple {'Brian', 'Xiao', 'Igor'} is 3.  
   The number of elements in this tuple {} is 0.
   ```
2. [15 marks] Consider the following Python code fragment:

```python
grade = input("Please, enter a grade between 0 and 100 : ")
if len(grade) == 0 :
    print("Nothing")
elif grade.isalpha() :
    print("Characters")
else :
    grade = int(grade)
    if grade < 0:
        print("Negative grade")
elif grade > 100:
    print("Grade > 100")
elif 0 <= grade < 50 :
    print("F")
elif 50 <= grade < 62 :
    print("D")
elif 62 <= grade < 75 :
    print("C")
elif 75 <= grade < 87 :
    print("B")
else :
    print("A")
```

In the table on the next page, list the minimum number of test cases we would need in order to completely test this Python code fragment. More specifically, describe the test data (Test Data) we use to test and what we expect to see on the interpreter shell screen. (Expected Results).

An example of a test case has been included in the table to illustrate what a test case and its expected results can be.
<table>
<thead>
<tr>
<th>Test Case #</th>
<th>Test Data</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I press the <em>Enter</em> key (I do not enter anything).</td>
<td>Nothing</td>
</tr>
<tr>
<td>2</td>
<td>“banana”</td>
<td>Characters</td>
</tr>
<tr>
<td>3</td>
<td>-23</td>
<td>Negative grade</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>71</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>94</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>150</td>
<td>Grade &gt; 100</td>
</tr>
</tbody>
</table>

NOTE: You had to enter a specific value (test data), for example: 0. It is not sufficient to say “I enter a number between (and including) 0 and 49.”
3. [20 marks] In francophone countries, many of the first names given to girls are produced by appending "tte" or "ette" to the end of some first names given to boys. For example,

- a boy can be named Pierre and a girl Pierreette,
- a boy can be named Jean and a girl Jeanette.

**Problem Statement:** On the next page, write a Python program called NameBot that produces a girl’s first name given a boy’s first name. It does this by appending "tte" or "ette" at the end of the given name depending on whether the given name ends with the letter "e" or not. Looking at the **Sample Runs** below, you see that when the user enters Pierre, NameBot appends "tte" to it because Pierre ends with the letter "e" and when the user enters Jean, NameBot appends "ette" to it because Jean does not end with the letter "e".

Here is a **Sample Run** (i.e., one execution of the program):

```
Please, enter name (enter 'X' or 'x' to stop):
---Nothing entered - Try again!
Please, enter name (enter 'X' or 'x' to stop): Pierre
***Name is Pierre and its feminine form is Pierreette.
Please, enter name (enter 'X' or 'x' to stop): Jean
***Name is Jean and its feminine form is Jeanette.
Please, enter name (enter 'X' or 'x' to stop): X
```

Looks like we may need a loop!

**Bonus Part:** If the name the user enters already finishes with "tte", then our program **NameBot** must print

"---BONUS - Already a girl's name - Try again!"

**Sample Run with Bonus Part:**

```
Please, enter name (enter 'X' or 'x' to stop):
---Nothing entered - Try again!
Please, enter name (enter 'X' or 'x' to stop): Pierre
***Name is Pierre and its feminine form is Pierreette.
Please, enter name (enter 'X' or 'x' to stop): Jean
***Name is Jean and its feminine form is Jeanette.
Please, enter name (enter 'X' or 'x' to stop): Claudette
---BONUS - Already a girl's name - Try again!
Please, enter name (enter 'X' or 'x' to stop): x
```

**Note:** There are no other invalid input aside from the ones used in the **Sample Runs** above (i.e., when the user does not enter anything, but presses the Enter key). In other words, the user will not enter a number or a mix of letters and digits when asked to enter a name.

If you find it useful to start answering this question by designing an algorithm, do so.
Put your answer to Question 3 here

Possible Solution:
https://repl.it/repls/GargantuanShyD11