SFU ID: ....
(not your student number!)

CMPT 120—QUIZ 2 (Mar. 7)

<u>DO NOT FORGET TO PUT YOUR SFU ID AT THE TOP LEFT OF EVERY PAGE.</u>
<u>THE SFU ID: is with letters (mine is "jlumbros").</u>

A few instructions:

- You may pick five out of the six exercises as long as they total to fifty minutes (or do all six);
- When you are done, **remain seated until the end of the quiz** (if you want some distraction, a challenge question has been placed at the end of this quiz);
- I know that other faculties ask for your student number (30114762...), but I am asking for your SFU ID (somename...); **ALL QUIZZES THAT HAVE A STUDENT NUMBER INSTEAD OF AN SFU ID WILL GET A -25.00 GRADE; NOT JUST 0.00, -25.00, CONSIDER YOURSELF FOREWARNED!!!!**
- Oh, but have fun. I believe in you! :-)

SFU ID: ....
(not your student number!)

EXERCISE 1: multiple-choice analysis of code snippets (10 minutes)

1) Suppose that `a`, `b` and `c` are **three strictly positive integer values that are correctly initialized** (to values that you don't know). Below are code snippets: analyze what they do, and write the correct letter in the box on the right.

```
if a > b:
  a = b
elif a > c:
  a = c
```

<u>After</u> execution of this code, the **value of a** is:
A) the maximum value of (the initial) `a`, `b` and `c`?
B) the minimum value of (the initial) `a`, `b` and `c`?
C) the median value of (the initial) `a`, `b` and `c`?
D) something else entirely?

```
if a > b:
  a = b
if a > c:
  a = c
```

<u>After</u> execution of this code, the **value of a** is:
A) the maximum value of (the initial) `a`, `b` and `c`?
B) the minimum value of (the initial) `a`, `b` and `c`?
C) the median value of (the initial) `a`, `b` and `c`?
D) something else entirely?

```
if (a-b)*(c-a) >= 0:
  a = a
elif (b-a)*(c-a)>= 0:
  a = b
else:
  a = c
```

<u>After</u> execution of this code, the **value of a** is:
A) the maximum value of (the initial) `a`, `b` and `c`?
B) the minimum value of (the initial) `a`, `b` and `c`?
C) the median value of (the initial) `a`, `b` and `c`?
D) something else entirely?

2) Suppose that `n` is **a strictly positive integer, correctly initialized**.

```
s = 0
n = n/2
for i in range(n+1):
  s = s + 2*i
```

<u>After</u> execution of this code, the **value of s** is:
A) the sum of all even integer from up to `n`?
B) the sum of all integers up to `2n`?
C) the value 2^n?
D) the value `1`?
E) something else entirely?

```
s = 0
for i in range(n+1):
  if s % 2 == 0:
    s = s + i
```

<u>After</u> execution of this code, the **value of s** is:
A) the sum of all even integer from up to `n`?
B) the sum of all integers up to `2n`?
C) the value 2^n?
D) the value `1`?
E) something else entirely?

```
s = 0
i = 0
while i < n:
  i = i + 2
  s = s + i
```

<u>After</u> execution of this code, the **value of s** is:
A) the sum of all even integer from up to `n`?
B) the sum of all integers up to `2n`?
C) the value 2^n?
D) the value `1`?
E) something else entirely?

SFU ID: ....
(not your student number!)

EXERCISE 2: execute a program (10 minutes)

It is strongly suggested that you execute this function step by step, on a piece of paper, as though you were running it with Python Tutor. If you show on this piece of paper that you are capable of running this function, even if you make a mistake on the final result, you will get partial marks.

```
def myfunc(first, last):
  acc = 0
  last = 10 - last
  first = 10 - first
  for i in range(last, first+1): #line was changed
    acc = acc + i
  for j in range(last, first+1): #line was changed
    acc = acc * 2
  return acc

mynum1 = myfunc(6, 7) + myfunc(7, 9)
mynum2 = myfunc(6, 9)

if mynum1 < mynum2:
  print "First wins"
elif mynum1 > mynum2:
  print "Second wins"
else:
  print "It's a tie"
```

At the end of the program:

- What is the value of `mynum1`? ....
- What is the value of `mynum2`? ....
- What does the program print out? ....

SFU ID: ....
(not your student number!)

EXERCISE 3: design test cases to show a function is not working (6 min.)

In this exercise, you are given some a problem, and a possible "solution"; your goal is to choose a test case (some values of the parameters) for which the expected value is different from the value computed by the function.

Example: The statement is: "*The function* `sum_range(n)` *must return the sum of integers from* `1` *to* `n` *(included), i.e., it must return* `1+2+...+(n-1)+n`."

The answer proposed below is incorrect, so I have checked the option "It doesn't work..." and I have given an example where the function returns an incorrect answer: if I take `n=3`, that is, if I call `sum_range(3)` with the code below, I get `3`, when the statement required me to return `6`.

```python
def sum_range(n):
  s = 0
  for i in range(n):
    s = s + i
  return s
```

[ ] This function works as expected.		
[X] It doesn't work, and here is a test case it fails:		
n	**should return**	**but returns**
3	1+2+3=6	1+2=3

Here the statement is: "*The function* `find_last(s, c)` *returns the last occurrence of character* `c` *in string* `s`, *and* `-1` *if* `c` *does not occur in* `s`."

Here is a first possible solution:

```python
def find_last(s, c):
  pos = -1
  for i in range(len(s)):
    if s[i] == c:
      pos = i
    else:
      pos = -1
  return pos
```

[ ] This function works as expected.			
[ ] It doesn't work, and here is a test case it fails:			
s	**c**	**should return**	**but returns**

Here is a second possible solution:

```python
def find_last(s, c):
  pos = -1
  for i in range(len(s)):
    if s[i] == c:
        pos = i
  return pos
```

[ ] This function works as expected.			
[ ] It doesn't work, and here is a test case it fails:			
s	**c**	**should return**	**but returns**

Here is a third possible solution:

```python
def find_last(s, c):
  pos = -1
  for i in range(len(s)):
    if s[i] == c:
      if pos == -1:
        pos = i
  return pos
```

[ ] This function works as expected.			
[ ] It doesn't work, and here is a test case it fails:			
s	**c**	**should return**	**but returns**

EXERCISE 4: palindromes, fill in the blanks (14 minutes)

In this exercise, we wish to recognize first whether a string is exactly palindromic, and then whether a string is approximately palindromic.

A string is **exactly palindromic** when it can be read exactly the same from both directions (left to right, and right to left). For instance, here are exactly palindromic words: "rotor", "noon", "kayak", "civic", "radar", these words can be reversed without making a difference.

A string is **approximately palindromic** if, when only alphabetical characters are kept (no punctuation, no spaces), and when all letters are put in lowercase, then it becomes exactly palindromic. Examples of such phrases:

- "A man, a plan, a canal - Panama!"
- "Never odd or even"
- "Cigar? Toss it in a can. It is so tragic."

1) The function `is_exact_palindrome(s)` must return `True` if the string `s` is an exact palindrome (that means the string is equal to its reverse exactly, with no changes to the string).

Fill in the blanks of this function with the possible replies below (the answer has been given for blank #6).

```
def is_exact_palindrome(s):
  1____________________
  while 2__________________:
    if 3___________________:
      4__________________
    5_________________
  6 return True
```

In blank:	I put option:
1	
2	
3	
4	
5	
6	A

A	`return True`	B	`s[i]==s[len(s)-i-1]`	C	`i < len(s)`
D	`s[i]!=s[-i-1]`	E	`i = 0`	F	`i < len(s)/2`
G	`s[i]!=s[len(s)-i-1]`	H	`s[i]==s[-i-1]`	I	`i < len(s)+1`
J	`flag = False`	K	`i = i + 1`	L	`i = 1`
M	`i = len(s)-1`	N	`s = s + i`	O	`new_s = ""`
P	`new_s= new_s + s[i]`	Q	`new_s= new_s + i`	R	`return False`

SFU ID: ....
(not your student number!)

2) Now **using the function `is_exact_palindrome(s)`**, we want to write a function `is_approximate_palindrome(s)`, which will determine if the string `s` is an approximate palindrome: that is, if it is a palindrome once any non-alphabetical characters have been removed, and all characters set to lowercase.

a) Write the function `normalize_string(s)`, which takes a string parameter `s`, and returns the same string, with all non-alphabetical characters removed, and all alphabetical characters set to lowercase.

....

....

....

....

....

....

....

....

....

....

....

....

b) Assuming both the two functions `is_exact_palindrome(s)` and `normalize_string(s)` exist and work properly as described above, fill-in the two blanks below with free-form answers

```
def is_approximate_palindrome(s):

  new_s = ...............................................

  return ................................................
```

(Note: there should not be more than two blanks filled, no extra lines.)

SFU ID: ....
(not your student number!)

EXERCISE 5: definitions (10 minutes)

1) Below is a list of Python keywords:

- circle the keywords that create a new blocks of code (where you have to indent the code that is below);
- describe shortly what the keyword is used for, or in what context you have been using it (no need to be detailed).

def	....
while	....
return	....
print	....
if	....

2) Describe the difference between a `for` loop and a `while` loop.

....

....

....

....

....

3) Provide a short example using a for loop (on the left), and rewrite that example using a while loop (on the right). For instance, if you have no ideas, the sum of integers between 1 and n.

....	....
....	....
....	....
....	....
....	....
....	....

EXERCISE 6: write your own function (10 minutes)

Write a function `print_lengths_of_words(s)`, which takes one string parameter `s`, and prints the size of each word contained in the string ("word" is considered to be any consecutive sequence of alphabetical characters).

For instance:

- `print_lengths_of_words("I did it for me. I liked it. I was good at it. And I was really... I was alive.")` will print (on a new line, or on the same line, it does not matter, the only thing that matters is that the numbers are all printed out): 1, 3, 2, 3, 2, 1, 5, 2, 1, 3, 4, 2, 2, 3, 1, 3, 6, 1, 3, 5.

- `print_lengths_of_words("Predator only hunts in tropical jungles. I assume. And desperately hope. ")` prints: 8, 4, 5, 2, 8, 7, 1, 6, 3, 11, 4.

SFU ID: ....
(not your student number!)

CHALLENGE QUESTION: number letter count
(only if you have time, and have completed the rest of the quiz)

If the numbers 1 to 5 are written out in words: “one, two, three, four, five,” then there are 3 + 3 + 5 + 4 + 4 = 19 letters used in total.

Question: If all the numbers from 1 to 1000 (one thousand) inclusive were written out in words, how many letters would be used?

Write Python code to calculate this.

Note: Do not count spaces or hyphens. For example, 342 (three hundred and forty-two) contains 23 letters and 115 (one hundred and fifteen) contains 20 letters. The use of "and" when writing out numbers is in compliance with British usage.

(Source: Project Euler, problem 17.)

SFU ID: ....
(not your student number!)