National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2021/2022

Extra Practice 2

Question 1

Trace the following code. **(4 marks)**

```
result = 0
for i in range(5):
    result += 1
print(result)

result2 = 0
for i in range(1, 5):
    result2 += 1
print(result2)
```

Question 2

Trace the following code. **(5 marks)**

```
result = 0
for i in range(1, 13, 3):
    if i % 2 == 0:
        i += 2
    else:
        result //= i
    result += i

print(result)
```

Question 3

Trace the following code. **(5 marks)**

```
a, b, c = "east", "easter", "easy"
a, b, c = c, a, b

if a < b:
    a, b = b, a
else:
    if b < c:
        a += b
b = a + c

print(a[:-1])
print(b[1:])
print(c[::-1])
```

Question 4

To play a game of bowling, we will store our results from each throw in an integer such as 1459. In this game, we will only play with 9 pins. 1459 means 1 pin is struck in the first shot, 4 pins in the second shot, 5 pins in the third shot and a strike in the last shot.

(a) Define a function `score` that takes in an integer and returns the total score of the game (1 pin = 1 point). Use an iterative approach. **(4 marks)**
Sample Tests:

```
>>> score(1459)
19
>>> score(999)
27
```

(b) What is the order of growth (in space and time) of your solution in part (a)? Explain your answer. **(2 marks)**

(c) Define a function `score_recursive` that does the same thing as in part a but in a **recursive** manner. **(4 marks)**

(d) What is the order of growth (in space and time) of your solution in part (c)? Explain your answer. **(2 marks)**

(e) Define a function `strike_count` and `strike_count_recursive` that takes in an integer and returns the total number of strikes in the game. Use iteration and recursion respectively. **(8 marks)**
Sample Tests:

```
>>> strike_count(919)
2
>>> strike_count(1234560)
0
>>> strike_count(9999)
4
```

(f) Now, each strike is going to be worth an extra 5 points each! Using your previously defined functions, define a new function **`score_improved`** that takes in an integer and returns the total score. **(4 marks)**
Sample Tests:

```
>>> score_improved(919)
29
>>> score_improved(1234)
10
>>> score_improved(12349)
24
```

Question 5

(a) We will define another maskify function to encrypt our password. Given a password of any length, we want to mask all the characters with "*". Define a function **`maskify`** that takes in a password as a string and returns the new masked word. Use **iteration**. **(4 marks)**
Sample Tests:

```
>>> maskify("password")
'********'
>>> maskify("121")
'***'
```

(b) State the time and space complexity of your solution. **(2 marks)**

(c) Do part (a) with recursion. **(4 marks)**

(d) State the time and space complexity of your solution. **(2 marks)**

(e) Now, we want to put an "*" sign in between all of the letters. Define a function **`slot`** that does this recursively. **(6 marks)**
Sample Tests:

```
>>> slot("pass")
'p*a*s*s'
>>> slot("123")
'1*2*3'
```

(f) We want to insert the "*" sign now into consecutive letters that are identical to each other only. Define a function **`advanced_slot`** that can do this recursively. **(6 marks)**
Sample Tests:

```
>>> advanced_slot("pass")
'pas*s'
>>> advanced_slot("aaaaba")
'a*a*a*aba'
```

Question 6

Trace the following code. **(4 marks)**

```
def weird_sum(n):
    if n == 0:
        return 0
    else:
        return n + weird_sum(n - 2)

print(weird_sum(5))
```

Question 7

Trace the following code. **(4 marks)**

```
for i in range(5):
    print(i)
    i += i
```

Question 8

You might have known about Fibonacci numbers before. But, have you known about Lucas numbers?

According to Wikipedia, we define Lucas numbers as follows

$$L_n = \begin{cases} 2 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ L_{n-1} + L_{n-2} & \text{if } n > 1 \end{cases}$$

(a) Define a **recursive** function `lucas` that takes in a *nonnegative integer* n and returns L_n. **(2 marks)**
Sample Tests:

```
>>> lucas(2)
1
>>> lucas(11)
123
```

(b) Do part (a) **iteratively**. **(3 marks)**

(c) **(Optional)** Define a function `lucas2` that takes in a *positive integer* n and returns

$$\frac{L_{n-1} + L_{n+1}}{5}$$

Try to call this function for small values of n. Does `lucas2` seem familiar to you?