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

Extra Practice 4

Help

Yes, we put these functions here so that you can refer to them easily.

```
def sum(term, a, next, b):
    if a > b:
        return 0
    else:
        return term(a) + sum(term, next(a), next, b)

def fold(op, f, n):
    if n == 0:
        return f(0)
    else:
        return op(f(n), fold(op, f, n-1))

def compose (f, g):
    return lambda x: f(g(x))

def thrice (f):
    return compose(compose(f, f), f)
```

Question 1

This question tests you about the left-right evaluation.

```
def new_if(predicate, then, otherwise):
    if predicate:
        then
    else:
        otherwise

def p(x):
    new_if(x > 5, print(x), p(x+1))

p(1) # what happens here?
```

Question 2

It's time for a simple function nesting! What's the output of the code below?

```
print((lambda x: lambda y: 2*x)(3)(4))
```

Question 3

Let's apply the same idea from Mission 3 :D

```
print(thrice(thrice)(lambda x: x-1)(27))
```

Question 4

Nested function calls. Can you do it?

```
foo = lambda y: lambda x: x(y)
add1 = lambda x: x+1

print(foo(add1(2))(foo)(add1))
```

Question 5

Nested variable scopes. Can you do it?

```
def foo(x):
    def bar(x, y):
        return lambda y: y(x)
    return lambda y: bar(x, y)

print(foo(lambda x: x**3)(lambda x: x**2)(lambda x: x)(4))
```

Question 6

Try to do each subquestion within **5 minutes**. You are to use the **`fold`** function in all of the following questions.

1. Define a function **`between`** that takes in a word as an input and returns a new string with "*" placed in between all consecutive words that are identical.
   Sample Tests:

   ```
   >>> between("happy")
   'hap*py'
   >>> between("oookayy")
   'o*o*okay*y'
   ```

2. Define a function **`check_vowel`** that takes in a word as an input and returns **`True`** if there is at least one vowel in the word or **`False`** otherwise. This is case-sensitive.

Sample Tests:

```
>>> check_vowel("qwertyjkl")
True
>>> check_vowel("482jfn")
False
```

3. We have a tuple that contains some integers. Define a function **largest** that takes in a tuple and returns the largest integer.
Sample Tests:

```
>>> largest((4,2,6,2,1))
6
>>> largest((0,0,0,0,0))
0
```

Question 7

Given a tuple containing numbers, find the number of combination of numbers that you can use from the tuple that can add up to a target number. Define this function **no_of_ways** that takes in a tuple and a target number, and return the number of ways you can hit this target. You can only use each number once. You do not need to use any higher-order functions for this.

Sample Tests:

```
>>> no_of_ways((4, 2, 5), 8)
0                                  # no way you can make the number 8
>>> no_of_ways((4, 2, 5, 3), 7)
2                                  # either 2 + 5 or 4 + 3 gives 7
>>> no_of_ways((4, 2, 5, 3, 5, 1), 5)
4                                  # either 4 + 1, 2 + 3, 5 or 5 gives 5
```

Question 8

Define function that returns the sum of cubes for numbers between 1 to n, using **accumulate**. Recall the definition of **accumulate** below.

$$a_1 = a, a_n \leq b$$

$$accumulate(\oplus, base, f, a, next, b) : (f(a_1) \oplus (f(a_2) \oplus (... \oplus (f(a_n) \oplus base)...)))$$

Question 9

Define the double factorial function using **accumulate**.

$$n!! = \begin{cases} (n)(n-2)(n-4)\cdots(4)(2) & \text{if } \frac{n}{2} \in \mathbb{Z}^+ \\ (n)(n-2)(n-4)\cdots(3)(1) & \text{if } \frac{n+1}{2} \in \mathbb{Z}^+ \end{cases}$$