National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2024/2025

Recitation 3
Higher Order Functions

Python

1. *lambda* - `lambda` *params:* `expression`
   Creates an anonymous function equivalent to:

   ```
   def function(params):
       return expression
   ```

Definitions

The following are two higher-order functions discussed in lecture:

```
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))
```

Note: it is not necessary to memorize these defintions, or even the names of these functions. Definitions of such functions (if they are used) will be given in an Appendix for examinations. What you need to be able to do is to read the definition for such functions and understand what they do and be able to use them.

Problems

1. Evaluate the return values of the following expressions:

   (a)
   ```
   x = 2
   def f():
       x = 5
       y = x + 5
       return x + y
   f() + x
   ```

(b)
```
x = 2
y = 3
def g(x):
    y = x + 5
    x = 7
    return x + y
g(y)+y
```

(c)
```
x = 4
def foo(x):
    return x(3)
foo(lambda x: x+1)
```

(d)
```
x = 5
def bar(x, y):
    return y(x)
bar(4, lambda x: x**2)
```

2. Write a function `my_sum` that computes the following sum, for $n \geq 1$:

$$1 \times 2 + 2 \times 3 + \cdots + n \times (n+1)$$

3. Is the function `my_sum` as defined above a recursive process or an iterative process? What is the order of growth in time and in space?

4. If your answer in Question 2 is a recursive process, re-write `my_sum` as an iterative process. If your answer in Question 2 is an iterative process, re-write `my_sum` as a recursive process. What is the new order of growth in time and space?

5. We can also define `my_sum` in terms of the higher-order function `sum`. Complete the definition of `my_sum` below. You cannot change the definition of `sum`; you may only call it with appropriate arguments.

```
def my_sum(n):
    return sum(<T1>, <T2>, <T3>, <T4>)

T1:


T2:


T3:


T4:
```

6. Suppose instead we define `my_sum` in terms of the higher-order function `fold`. Complete the definition of `my_sum` below.

```
def my_sum(n):
    return fold(<T1>, <T2>, <T3>)

T1:


T2:


T3:
```

7. Write an iterative version of `sum`.

8. **Homework:** Write an iterative version of `fold`.