National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester II, 2016/2017

Tutorial 3
Recursion, Iteration & Orders of Growth

1. Draw the tree illustrating the process generated by the `cc(amount,d)` function given in the lecture, in making change for 11 cents. What are the orders of growth of the space and number of steps used by this process as the `amount` to be changed increases?

2. A function f is defined by the rule that $f(n) = n$ if $n < 3$ and $f(n) = f(n-1) + 2f(n-2) + 3f(n-3)$ if $n \geq 3$.

   (a) Write a function that computes f by means of a recursive process.

   (b) Write a function that computes f by means of an iterative process.

   State the order of growth for Time and Space complexity for both implementations.

3. Write a function `is_fib(n)` that returns `True` if n is a Fibonacci number, and `False` otherwise. What is the order of growth in terms of time and space for the function that you wrote? Explain.

4. Recall the `taxi_fare` example given in lecture 2.

```
stage1 = 1000
stage2 = 10000
start_fare = 3.0
increment = 0.22
block1 = 400
block2 = 350

def taxi_fare(distance): # distance in metres
    if distance <= stage1:
        return start_fare
    elif distance <= stage2:
        return start_fare + (increment *
                             ceil((distance - stage1) / block1))
    else:
        return taxi_fare(stage2) + (increment *
                                    ceil((distance - stage2) / block2))
```

   We would like to avoid the use of global variables, i.e. the variables defined outside the function. In the lectures, we talked about variable scope within functions. We are allowed to define an inner function that makes use of a variable bound within an outer function. An example from the lectures is this:

```
def hypotenuse(a,b):
    def sum_of_squares():
        return square(a)+square(b)
    return sqrt(sum_of_squares())
```

What if we were to return the inner function instead of just using it locally? The returned function would have access to the variables bound when it was returned.

Define a function `make_fare` that takes as arguments `stage1`, `stage2`, `start_fare`, `increment`, `block1`, `block2` and returns a function that calculates the `taxi_fare` using those values.

```
def make_fare(stage1, stage2, start_fare, increment, block1, block2):
    "Return a function that calculates the taxi_fare using
     the values given."

>>> comfort_fare = make_fare(1000, 10000, 3.0, 0.22, 400, 350)
>>> comfort_fare(3500)
4.54
```