Discussion 11: HOFs, Lambda Functions, Tree Recursion

Lambda Functions

1. Write a lambda function called f that takes in a number and outputs that number squared.

   \[
   f = \ldots
   \]

2. Now, use a list comprehension and your lambda function f to return a list the squares of all numbers between 1-5, inclusive.

Functions as Data

1. What would the Python interpreter display for the following lines of code? If you believe a line errors, just write “Error.” Assume that the lines are executed independently, not sequentially.

   ```python
   >>> f1 = lambda x: x + x
   >>> f2 = lambda x: x > 9
   >>> [f(10) for f in [f1, f2]]
   ```

   ```python
   >>> f = lambda x: lambda: x + x
   >>> f(2)
   ```

   ```python
   >>> y = 3
   >>> f = lambda x: lambda: x + y
   >>> f(2)()
   ```

   ```python
   >>> g = lambda y: x + y
   >>> g(2)
   ```

2. Now, continue the exercise, instead assuming that the lines are executed sequentially.

   ```python
   >>> def make_adder(x):
   ...     def inner(y):
   ...         return x + y
   ...     return inner
   >>> make_adder(5)
   ```

   ```python
   >>> make_adder(5)(6)
   ```
3. Write a function called `functionList` that takes in a list of functions, `functions`, and a number, `n`, and returns a list of the results of calling each function on `n`.

```python
>>> functionList([lambda x: x + x, lambda x: x * x], 4)
[8, 16]
```

4. Write a recursive function called `recursiveSum` that takes in a function `func` and a number `n`, and returns the summed results of `func` applied from 1 to `n`.

```python
>>> recursiveSum(lambda x: x * x, 3)
14 # 3*3 + 2*2 + 1*1
```
Tree Recursion

1. The Fibonacci sequence is a sequence of numbers where each number is the sum of the previous two. Here is the start of the Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, ...

In the space below, write the function fib(n) that returns the nth Fibonacci number in the sequence, assuming the first one is $n = 0$.

What is the runtime of this function? ____________________________________________

2. We find ourselves at the bottom of a staircase with num_steps steps. We can either climb the stairs one at a time or two at a time (or a mix of the two). Fill in the function below to return the number of ways you can climb the staircase.

```python
def climb_staircase(num_steps):
    if num_steps == 0:
        return ________________________________
    elif num_steps < 0:
        return ________________________________
    else:
        return ________________________________
```

3. Now, when we are climbing the staircase, we can take any from 1 to max_steps number of steps at a time (not just 1 or 2). Fill in the blanks below to rewrite climb_staircase to return the number of ways you can now climb the staircase.

```python
def climb_staircase(num_steps, max_steps):
    if num_steps == 0:
        return ________________________________
    elif num_steps < 0:
        return ________________________________
    else:
        return ________________________________