Midterm Exam #2

CS97: Principles and Practices of Computing

Wednesday, November 15, 2017

1.	2.
3.	4.

Name:_____

ID:_____

Rules of the game:

- **Write your name and ID number above.**
- The exam is closed-book and closed-notes.
- Please write your answers directly on the exam. Do not turn in anything else.
- The exam ends promptly at 3:50pm.
- Read questions carefully. Understand a question before you start writing.
- Relax!

1. (2 points each) The *Fibonacci sequence* of nonnegative numbers is defined as follows. The 0th Fibonacci number is 0, the 1st Fibonacci number is 1, and the nth Fibonacci number (where $n > 1$) is the sum of the previous two Fibonacci numbers.

   Here is a Python function that returns the nth number in the Fibonacci sequence:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

   (a) What is the result of `fib(4)`?

   3

   (b) What are the two argument integers to the first addition operation (i.e., execution of the `+` operator) that happens during the execution of `fib(4)`? Provide the integers in order from left to right.

   1 and 0

   (c) How many times is `fib(1)` executed during the execution of `fib(4)`?

   3

2. (5 points) Implement a function `makeList` that takes a function `f` and a nonnegative integer `n` and returns the list `[f(0),...,f(n-1)]`. For example, `makeList(lambda x: 2*x, 5)` returns `[0, 2, 4, 6, 8]` and `makeList(lambda x: 2*x, 0)` returns `[]`.

```
def makeList(f, n):
    if n == 0:
        return []
    else:
        return makeList(f, n-1) + [f(n-1)]
```

3. (2 points each) **Circle all answers that are true for each question.**

   (a) Which functions *always* return a list of length less than that of the argument list?

      i. `map`
      ii. `filter`
      iii. `reduce`
      iv. none of the above

      iv

   (b) Which functions *always* return a list of the same type of elements as the argument list (e.g., given an integer list, the function will always return an integer list)?

      i. `map`
      ii. `filter`
      iii. `reduce`
      iv. none of the above

      ii

   (c) Which functions *always* return a list?

      i. `map`
      ii. `filter`
      iii. `reduce`
      iv. none of the above

      i and ii

   (d) Which functions *never* return a list?

      i. `map`
      ii. `filter`
      iii. `reduce`
      iv. none of the above

      iv

4. (5 points) Implement a function `swapPairs` that takes a list of pairs and returns an identical list but with the elements of each pair swapped. For example,

```
swapPairs([[1, "one"], [2, "two"], [3, "three"]])
```

returns `[["one", 1], ["two", 2], ["three", 3]]`.

You may not use recursion for this problem. Instead, make appropriate use of `map`, `filter`, and/or `reduce`.

```
def swapPairs(alist):
    return list(map(lambda p: [p[1], p[0]], alist))
```