CSCI 2113 Lab 10

April 17 2017

Goals

1. Practice write recursive functions
2. Practice converting recursive functions into tail recursive functions

Activity

1. Download Lab10.scala file. And add your implementations to the file.
2. Study the factorial function and how @tailrec is used. (Notice the import line at the top.)
3. Reimplement the functions from Lab 8 using recursion

```
def paranMatch(chars: Array[Char]): Boolean = ???
```

Hint1: you might have to create a nested function that keeps track of the counter
Hint2: you might need extra array methods such as .head, .tail and .isEmpty
You can read about the methods here: scala-lang.org/api/current/scala/Array.html

4. Implement the following 3 functions using recursion

```
//Fibonacci number is a sequence of numbers where every number
//after the first two is the sum of the two preceding ones.
//https://en.wikipedia.org/wiki/Fibonacci_number
//First two numbers are always 0 then 1.
//This function should return the nth fibonacci number
//Example
//  fib(0) == 0, fib(1) == 1, fib(2) == 1, fib(3) == 2, fib(4) ==
3
def fib(n: Int): Int = ???

//Palidrome sequence of characters which
//reads the same backward as forward, like "madam" or "racecar"
//https://en.wikipedia.org/wiki/Palindrome
//Write a function that determines if an array of characters are
//Palidrome.
//Examples:
//  isPalindrome("Hello") == false
//  isPalindrome("racecar") == true
//  isPalindrome("Racecar") == false
//  isPalindrome("())(") == true
//  isPalindrome("()()") == false
def isPalindrome(chars: Array[Char]): Boolean = ???
```

```
//Write a function that calculates power of x to the nth power.
//In other words, multiply x by itself n times.
//power(2, 3) is equivalent tot 2 * 2 * 2
def power(x: BigInt, n: BigInt): BigInt = ???
```

3. Determine which functions in #1 and #2 are tail recursion or not. If they are not, rewrite the function as tail recursion.

Assignment

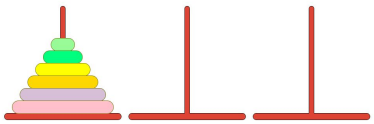
1. Read about Tower of Hanoi here: https://en.wikipedia.org/wiki/Tower_of_Hanoi
2. Try playing the game here: https://www.mathsisfun.com/games/towerofhanoi.html
3. Now, write a function that determines that minimal number of moves required to solve a tower with height of n.

```
def hanoi(n: BigInt): BigInt = ???
```

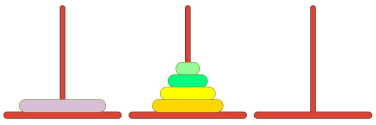
Hint:

If the height is 1 then only move required is to move the single piece to the goal. Therefore, hanoi(1) == 1

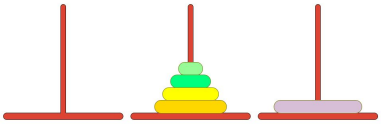
If the height is n that's greater than one then we can assume that we know the minimal number less than n. So we'll use those moves to move n-1 pieces to the middle peg. Then move the biggest piece to the goal peg. Then use that same strategy that we used to move n-1 pieces to the goal peg.



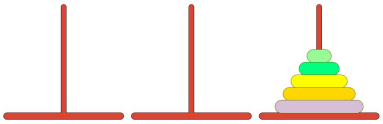
Start with height of n



Move n -1 pieces using hanoi(n-1) moves



Move the nth largest piece to the goal peg



Move the rest of the pieces using hanoi(n-1) moves

4. If your hanoi function is not tail recursive, convert it being tail recursive.
5. Make sure you do NOT use var or any of the mutable methods in Array object. You will receive partial credit if your function has side-effects.
6. Submit Lab10.scal file.
7. You will be graded on all 5 functions and to receive full credit, all your functions need to be tail recursive.