22C:16 Practice Problem Set 10

Morning Section: Complete before Tuesday, 4-30-2013

Evening Section: Complete before Monday, 4-29-2013

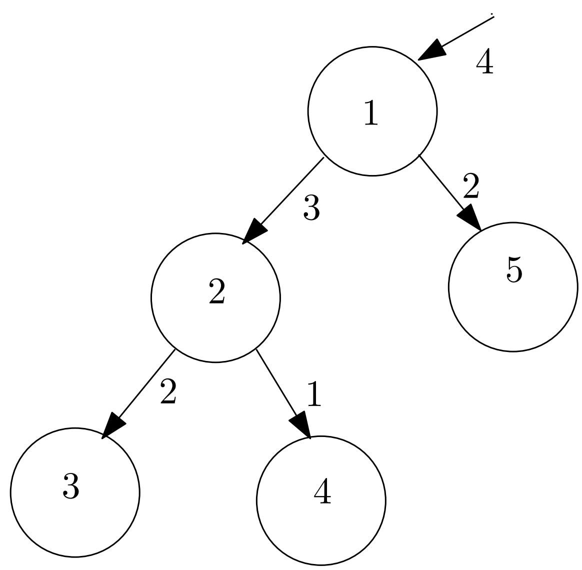
These practice problems are all on recursion.

1. This question is about the `fibonacci` function shown below.

```
def fibonacci(n):
    if n == 1 or n == 2:
        return 1

    answer = fibonacci(n-1) + fibonacci(n-2)
    return answer
```

   (a) Here is a picture that shows all the function calls that are made when we call `fibonacci(4)`. Specifically, the picture shows the parameters being sent into each function call (next to each arrow) and also the order in which the functions are called (inside each circle). Draw a larger version of this picture for `fibonacci(6)`. each circle



   (b) What output does the function produce, if we insert a `print n` statement as the very first line of the function and call `fibonacci(6)`. You should solve the problem by hand and not by running this function on a computer.

   (c) What output does the function produce, if we insert a `print n` statement as the second-last line of the function (just above the `return` statement) and call `fibonacci(6)`. You should solve the problem by hand and not by running this function on a computer.

2. Consider the recursive implementation of the function `power` that we discussed in class (see posted code).

   (a) What output does the function produce, if we insert a `print n` statement as the very first line of the function and execute the function call `power(2, 573)`.

(b) How many multiplications are performed by the function when we make the function call `power(3, 33)`?

3. Consider the recursive function `binarySearch`. Let `L` be the list `3, 5, 6, 11, 100, 123, 160, 178, 1000]`.

   (a) What output would the function produce if we inserted a **`print left, right`** as the first line of the function and called `binarySearch(L, 7, 0, 8)`? Also, write down the sequence of numbers that 7 is compared with during the course of the functin execution.

   (b) Same question as (a), but with function call `binarySearch(L, 1000, 0, 8)`.

4. Write a *recursive* function called **`recursiveLinearSearch`** with the following function header:

   `def recursiveLinearSearch(L, k, left, right)`

   This function searches the slice `L[left:right+1]` of the list `L` for the value `k` are returns `True` if the value is found; and `False` otherwise. Clearly, identify the base case(s) and recursive case(s). You cannot assume that the list `L` is sorted and hence you cannot do binary search.

5. Write a *recursive* function called **`isSorted`** with the following function header:

   `def isSorted(L, left, right)`

   This function determines if the slice `L[left:right+1]` of the list `L` is sorted in ascending order. If so, the function returns `True`; otherwise, the function returns `False`.

6. Write a *recursive* function for converting integers in decimal to equivalent binary numbers. Your function should use the following algorithm.

   > If the given integer n is even, then compute the binary equivalent of $n/2$ and append "0" to it. If n if odd, compute the binary equivalent of $n/2$ and append a "1" to it.

   I have deliberately left out any description of the base cases in the above pseudocode. Use the following function header:

   `def recursiveI2B(n):`
