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) 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.

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

2. Insert the statement

   `print L[first:last+1]`

   as the first line of the function `generalMergeSort` (i.e., just before the comment line on "Base case"). Write down the output produced by the function call

   `mergeSort([3, 6, 4, 11, -4])`

3. Insert the statement

   `print L[first:last+1]`

   as the last statement of the `merge` function. Write down the output produced by the function call

   `mergeSort([3, 6, 14, 1, 4])`

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):
```

7. You are given a list `L` of numbers and your task is to write a recursive function to determine the minimum number in `L`. Use the following function header:

```
def minimum(L):
```

For example, if `L` is `[21, 3, 7, 67, 19, 210, 21]` then the function should return 3. Of course this problem can be solved non-recursively, but you will not receive any credit for a non-recursive solution, even if it is correct. And, by the way, do not forget to specify the base cases.

Hint: To find the minimum number in `L` first find the minimum number in the sublist of `L` that excludes the first element. Then you just have to compare this with the first element in `L` to determine the answer.
