INTRODUCTION TO ALGORITHMS (CELEN086)

EXTRA PRACTICE PROBLEMS (3)

TOPIC: ***Helper functions, Lists***

Note: ALL algorithms must come with proper headings. Also, trace your algorithms!!

1. Write a recursive algorithm `numDigits(n)` that takes a positive integer and counts the number of its digits. For example: `numDigits(11)=2,`   `numDigits(54800)=5,`   `numDigits(8)=1`

2. Write a recursive algorithm called `mySqrt(n)` that takes a positive integer *n* and returns the largest positive integer *m* such that $m \times m \leq n$. You should use a helper function `mySqrtHelper().` Trace your algorithm for some *n* values. For example: `mySqrt(20)=4,`   `mySqrt(100)=10`

3. Now, use the function `mySqrt(n)` inside a new algorithm `isPrime(p)` that takes a positive integer and returns TRUE if it is prime and FALSE otherwise. You will again need a `isPrimeHelper()` function.

   Compare this new algorithm to the one presented to you in Lecture 4. Which one is faster and why? Trace both algorithms for `p=1001` and see which one gives the result faster?

4. Write a recursive algorithm `reverse(list)` that takes a list and returns a new list with the elements placed in reversed order. For example: `reverse([7,0,9,3,4,5])=[5,4,3,9,0,7].` You should write a helper function `reverseHelper().` Trace your algorithm for the list in the given example.

5. Write a recursive algorithm `num2list(n)` that takes a positive integer and returns a list that contains the digits of the input integer in the correct order. For example: `num2list(6)=[6], num2list(35181)=[3,5,1,8,1].` **HINT:** You may need to call the `reverse()` function!

6. Fibonacci numbers is a well-known sequence in mathematics. The following is a Fibonacci sequence:

   0, 1, 1, 2, 3, 5, 8, 13, 21, …

   Every number in the sequence is the sum of previous two numbers. The sequence begins with 0 and 1 and then progresses forward.

   Write a recursive algorithm `fiboList(n)` that takes a positive integer and returns a list whose elements are the first *n* Fibonacci numbers. For example: `fiboList(5)=[0,1,1,2,3].` You should write a helper function `fiboListHelper()` and you can call `Fibo(n)` function (from Problem Set 2) inside your helper function.

7. Write a recursive algorithm `splitEO(list)` that takes a list of integers and splits it into two new lists such that one contains all the odd-positioned elements of the input list and the other list contains all the even-positioned elements. For example: `splitEO([2,5,6,8,7,3,4,0])=([2,6,7,4],[5,8,3,0])`*(note the order in which elements appear is crucial)*. You can write two helper functions, one for extracting the even-positioned numbers and one for the odd-positioned numbers. ***However, it is possible to write an algorithm without the need for helper functions.*** Trace your algorithm for the example above.