These practice problems are all on recursion.

1. This question is about the \texttt{fibonacci} function shown below.

\begin{verbatim}
def fibonacci(n):
    if n == 1 or n == 2:
        return 1
    answer = fibonacci(n-1) + fibonacci(n-2)
    return answer
\end{verbatim}

(a) Here is a picture that shows all the function calls that are made when we call \texttt{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 \texttt{fibonacci(6)}. Each circle

\begin{center}
\begin{tikzpicture}
    \node (1) at (0,5) {1} ;
    \node (2) at (0,2.5) {2} ;
    \node (3) at (-1,0) {3} ;
    \node (4) at (1,0) {4} ;
    \node (5) at (0,-2.5) {5} ;

    \draw[-stealth] (1) -- (2) node [midway, above] {3} ;
    \draw[-stealth] (2) -- (3) node [midway, above] {2} ;
    \draw[-stealth] (2) -- (4) node [midway, above] {1} ;
    \draw[-stealth] (1) -- (5) node [midway, above] {4} ;
\end{tikzpicture}
\end{center}

(b) What output does the function produce, if we insert a \texttt{print n} statement as the very first line of the function and call \texttt{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 \texttt{print n} statement as the second-last line of the function (just above the \texttt{return} statement) and call \texttt{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 \texttt{power} that we discussed in class (see posted code).

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

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

3. Consider the recursive function \texttt{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 \texttt{print left, right} as the first line of the function and called \texttt{binarySearch(L, 7, 0, 8)}? Also, write down the sequence of numbers that 7 is compared with during the course of the function execution.

(b) Same question as (a), but with function call \texttt{binarySearch(L, 1000, 0, 8)}.

4. Write a \textit{recursive} function called \texttt{recursiveLinearSearch} with the following function header:

\begin{verbatim}
def recursiveLinearSearch(L, k, left, right)
\end{verbatim}

This function searches the slice \( L[left:right+1] \) of the list \( L \) for the value \( k \) and returns \texttt{True} if the value is found; and \texttt{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 \textit{recursive} function called \texttt{isSorted} with the following function header:

\begin{verbatim}
def isSorted(L, left, right)
\end{verbatim}

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

6. Write a \textit{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:

\begin{verbatim}
def recursiveI2B(n):
\end{verbatim}