Virtual Lecture Notes (Part 2)

Problem: Write a method to count the number of digits in a non-negative, base ten integer value *n*.

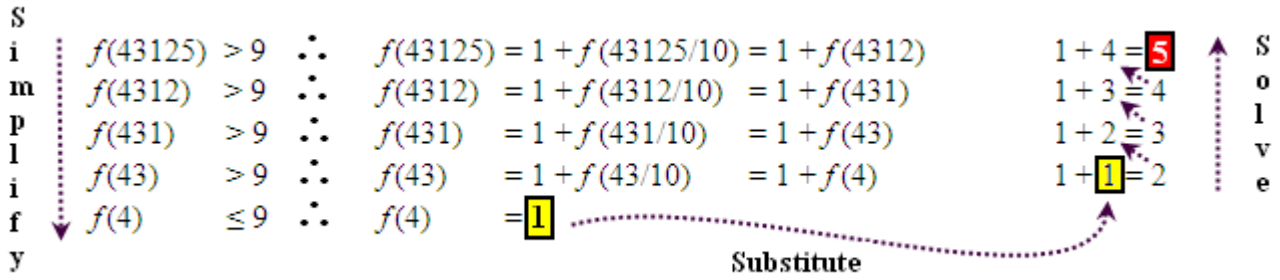
Example: The integer 43125 has five digits.

Algorithm: When the number is less than or equal to 9, n contains a single digit. Otherwise, count the number of times that n can be divided by ten, then finally add one to account for the least significant digit.

Piecewise Function: The algorithm for counting the number of digits in a non-negative, base ten integer value *n* is as follows.

$$f(n) = \begin{cases} 1 + f(n/10) & \text{if } n > 9 \\ 1 & \text{if } n <= 9 \end{cases}$$

Evaluation: Repeatedly dividing by 10 eliminates one position in *n* each time the method calls itself. Use the S-S-S Strategy to evaluate the function when n = 43125.



Implementation: Write the recursive method **`numberOfDigits()`** to count the number of digits in a non-negative, base ten integer value *n*. Notice that the recursive method design template has been used.

```
//precondition: n positive integer
int numberOfDigits(int n)
{
        if (n <= 9)
        {
             return 1;
        }
        else
        {
              return 1 + numberOfDigits(n/10);
        }
}
```

Method Trace: The trace of the **`numberOfDigits()`** method is shown below. The stack is represented on the left, showing the first unfinished method call at the bottom and the result of the base case at the top. After the base case is reached, evaluation of unfinished method calls resumes at the top of the stack and proceeds to the bottom.

Recursive call 5:	f(4)	= 1		return 1
Recursive call 4:	f(43)	= 1 + f(43/10)	= 1 + f(4)	return 1 + 1 = 2
Recursive call 3:	f(431)	= 1 + f(431/10)	= 1 + f(43)	return 1 + 2 = 3
Recursive call 2:	f(4312)	= 1 + f(4312/10)	= 1 + f(431)	return 1 + 3 = 4
Recursive call 1:	f(43125)	= 1 + f(43125/10)	= 1 + f(4312)	return 1 + 4 = 5
		The Stack		Evaluation

Do not let this confuse you when compared to the diagram shown in the Evaluation paragraph above. Just remember it is like stacking plates; the last one added to the stack has to be processed first when evaluation resumes. That is recursion!

Practice: Be sure to type this method into BlueJ and run it with several different values of n. Do not forget to add a main method.