Problem 3

Algorithm recursiveFactorial (n) *Count of operations*
Input: a non-negative integer *n*
Output: *n!*

if (n = 0 | n = 1) then	*3*
return 1	*1*
return n * recursiveFactorial(n – 1)	*4 (n – 1)*

a. Guessing method:
T(0) = 4
T(1) = 4
T(2) = 4 + T(1) = 4 + {4}
T(3) = 4 + T(2) = 4 + {4 + 4}
T(4) = 4 + T(3) = 4 + {4 + 4 + 4}
T(5) = 4 + T(4) = 4 + {4 + 4 + 4 + 4}

T(n) = 4 + T(n – 1) = 4 + 4 (n – 1)

Asymptotic running time:

$$\lim_{n\to\infty} \frac{4+4(n-1)}{n} = \lim_{n\to\infty} \frac{\frac{4}{n}+4(1-\frac{1}{n})}{1} = 0+4(1-0) = 4 \to T(n)\ is\ O(n).$$

b. Proof of algorithm correctness:

1. It has a base case, i.e. a line of code that executes without calling the function recursively. This is the line: `if (n = 0 | n = 1) then return 1`.
   Also, since the recursion line "`return n * recursiveFactorial (n – 1)`" subtracts "1" with each call, then it will eventually lead to the base case *n = 1*.
2. The base cases mentioned above return "1", which is correct by definition of the factorial function.
3. Assume the call to `recursiveFactorial (n – 1)` will return a correct value, then we try to prove that the call to `recursiveFactorial (n)` is correct, too: According to the algorithm above, the call to `recursiveFactorial (n)` is equal to `n*recursiveFactorial (n – 1)`, which is equal to n!.
4. From the three points above, we have the proof that the proposed `recursiveFactorial` **algorithm is correct.**