Section 4.4

1. Tracing factorial(n) for n=4:
   \( \text{factorial}(4) = 4 \) \( \text{factorial}(3) = 4.3 \) \( \text{factorial}(2) = 4.3.2 \) \( \text{factorial}(1) = 4.3.2.1 \) \( \text{factorial}(0) = 4.3.2.1.1 \)

5. Tracing the robot walking algorithm for n = 4
   \( \text{walk}(4) = \text{walk}(3) + \text{walk}(2) = [\text{walk}(2) + \text{walk}(1)] + 2 = 2 + 1 + 2 = 5 \)

7. Showing the robot walking algorithm is correct:
The algorithm is mean to compute the walk(n) function as defined at the top of the algorithm.
   If \( n = 1 \), then at the first if statement the algorithm will return 1 and we exit the algorithm.
   If \( n = 2 \), then at the first if statement the algorithm will return 2 and we exit the algorithm.
   If \( n \geq 3 \) the algorithm will return walk(n-1) + walk(n-2) as required. This will call on the same algorithm to find walk(n-1) and walk(n-2).
   Hence for all \( n \geq 1 \) the algorithm returns the correct value.

9. \( \text{sum}(s,n) \) {
   if \( s == 1 \) return \( \text{sum}(1) = 1 \)
   return \( \text{sum}(n) = \text{sum}(n - 1) + n \}

Proof: base case – if \( n = 1 \) \( \text{sum}(1) = 1 \)
General case – suppose \( \text{sum}(k) = 1 + 2 + \ldots + k \)
Then the algorithm computes \( \text{sum}(k+1) = \text{sum}(k) + 2(k+1) = 1 + 2 + \ldots + k + (k+1) \).
Hence the algorithm returns the correct value.

10. \( \text{sum}(s,n) \) {
     if \( s == 1 \) return \( \text{sum}(1) = 2 \)
     return \( \text{sum}(n) = \text{sum}(n - 1) + 2n \}

Proof: base case – if \( n = 1 \) \( \text{sum}(1) = 2 \)
General case – suppose \( \text{sum}(k) = 2 + 4 + \ldots + 2k \)
Then the algorithm computes \( \text{sum}(k+1) = \text{sum}(k) + 2(k+1) = 2 + 4 + \ldots + 2k + 2(k+1) \).
Hence the algorithm returns the correct value.

11. We discussed this in class. See notes for details.
   \( \text{walk}(n) = \begin{cases} 
   1 & \text{if } n = 1 \\
   2 & \text{if } n = 2 \\
   4 & \text{if } n = 3 \\
   \text{walk}(n - 1) + \text{walk}(n - 2) + \text{walk}(n - 3) & \text{if } n \geq 4 
   \end{cases} \)

We need to write an algorithm that computes this function.
walk(n) {
    if (n == 1 || n == 2)
        return n
    if (n == 3)
        return 4
    return walk(n - 1) + walk(n - 2) + walk(n - 3)
}

The proof is similar to the proof of the original walk algorithm.