# 1 Recurrence relations; Analysis of recursive algorithms

## 1.1 Recurrence relations

Solve the following recurrence relations.

**a.** $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
**b.** $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
**c.** $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
**d.** $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)
**e.** $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

## 1.2 Recursive algorithm

Consider the following recursive algorithm for computing the sum of the first $n$ cubes: $S(n) = 1^3 + 2^3 + \cdots + n^3$.

**ALGORITHM** $S(n)$
//Input: A positive integer $n$
//Output: The sum of the first $n$ cubes
**if** $n = 1$ **return** 1
**else return** $S(n-1) + n * n * n$

a. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.

b. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this sum?

### 1.3 Another recursive algorithm

Consider the following recursive algorithm.

**ALGORITHM** *Riddle*($A[0..n-1]$)
  //Input: An array $A[0..n-1]$ of real numbers
  **if** $n = 1$ **return** $A[0]$
  **else** $temp \leftarrow Riddle(A[0..n-2])$
    **if** $temp \leq A[n-1]$ **return** $temp$
    **else return** $A[n-1]$

a. What does this algorithm compute?

b. Set up a recurrence relation for the algorithm's basic operation count and solve it.