Introductory Lisp Exercises

Write some Lisp code to:

1. Find the length of a list
2. Reverse a list
3. Reverse a list and all its sublists
4. Take a list, and return a list containing all prefix sublists: `(a b c) -> ((a) (a b) (a b c))`
5. Flattens a list, e.g., `((a b) ((c)) d) -> (a b c d)`
6. Take two lists and return their Cartesian product, e.g., `(a b c)` and `(1 2) ->` `((a 1) (a 2) (b 1) (b 2) (c 1) (c 2))`
7. Compute the sum of all integers from 1 to n
8. Find the largest value in a list of numbers
9. Find the middle value in a list of numbers
10. Determine whether some value is a member of a list
11. Determine whether some value is a member of a list or any of its sublists
12. Take a value and a list and returns the list with all occurences of that value removed
13. Take two values and a list and returns the list with all occurences of the first value replaced by the second
14. Take a list and return a new list with all duplicated values removed
15. Define `equal` using only `eq`, `=`, `string=` and similar
16. Bubblesort a list of numbers
17. Quicksort a list of numbers
18. Sort a list of strings lexicographically
19. Take a list of values and a comparison function, and return the values sorted according to the function, e.g., `(3 2 1)` and `< -> (1 2 3)`
20. Take a predicate function and a list and return a new list of those elements of the list for which the predicate returns true
21. Take a function and a list and apply that function to each value in the list, returning a new list of the results
22. Return true is a list is a palindrome, e.g., `(1 2 a b a 2 1)` (do not use `reverse`)
23. Compute the gcd of two integers
24. Implement rationals as pairs of integers, and implement `+`, `-`, `*` and `/`
25. Evaluate simple prefix arithmetic expressions, e.g., `(+ (* 2 3) 5)`
26. Evaluate simple infix arithmetic expressions, e.g., `((2 * 3) + 5)`

27. Take two functions f and g of one argument and return a function of one argument that is the function composite $f \circ g$

28. Take a function f of two arguments and returns a function f_c of one argument that takes a value and returns a function that, when applied to a second value returns the value that f applied to the two values would have returned; i.e., $f_c(x)(y) = f(x, y)$

29. Take a function like f_c above, and returns a function f of two arguments such that $f(x, y) = f_c(x)(y)$

30. Take a function f of one argument and a non-negative integer n, and returns a function f_n such that f_n is f applied n times; i.e., $f_n(x) = f(f(\ldots(f(x))\ldots))$

Course Web page: `http://people.bath.ac.uk/masrjb/CourseNotes/cm20214.html`