Exercicis de LISP

1. Escriure la funció exponenciació.
   (exp 2 3) →8

```
(defun exp (m n)
   (cond ((= n 0) 1)
         (t (* m (exp m (- n 1))))))
```

2. Escriure la funció fibonacci que retorna l'enèsim valor d'aquesta sèrie.
   (fibonacci 5) →8

```
(defun fib (n)
   (cond ((= n 1) 1)
         ((= n 2) 1)
         (t (+ (fib (- n 1)) (fib (- n 2))))))
```

3. Escriure una funció que donada una llista retorni la mateixa llista sense els n primers elements.
   (borraprimers 3 '(a e i o u)) → (o u)

```
(defun borraprimers (n l)
   (cond ((= n 0) l)
         (t (borraprimers (- n 1) (cdr l)))))
```

4. Escriure una funció que donada una llista torni els primers n elements.
   (primers 3 '(a e i o u)) → (a e i)

```
(defun primers (n l)
   (cond ((= n 0) nil)
          (t (cons (car l) (primers (- n 1) (cdr l))))))
```

5. Escriure les funcions necessàries per insertar un element dins una llista:
   a) a l'esquerra d'un element donat
      (inserta-esquerra 'e 'a '(o r d e n a d o r)) → (o r d a e n a d o r)

```
(defun inserta-esquerra (dequi que l)
   (cond ((null l) nil)
         ((equal (car l) dequi) (cons que l))
         (t (cons (car l)
            (inserta-esquerra dequi que (cdr l))))))
```

   b) a la dreta d'un element donat
      (inserta-dreta 'e 'a '(o r d e n a d o r)) → (o r d e a n a d o r)

```
(defun inserta-dreta (dequi que l)
   (cond ((null l) nil)
         ((equal (car l) dequi)
                 (cons (car l) (cons que (cdr l))))
```

```
          (t (cons (car l)
                   (inserta-dreta dequi que (cdr l))))))
```

c) al mateix lloc d'un element donat
(substituir 'e 'a '(o r d e n a d o r)) → (o r d a n a d o r)

```
(defun substituir (qui perque l)
   (cond ((null l) nil)
         ((equal (car l) qui)
                 (cons perque (cdr l)))
         (t (cons (car l)
                  (substituir qui perque (cdr l))))))
```

6. Divisió sencera de M entre N (tots dos positius). Dóna el número de vegades que N està contingut dins M.
Ex. (dividir 9 2) → 4

```
(defun dividir (m n)
  (cond ((< m n) 0)
        (t (+ 1 (dividir (- m n) n)))))
```

7. Escriure una funció recursiva "senars" que donada una llista de números retorna una llista amb tots els números parells eliminats. Escriure també la funció parell a partir de dividir.
Ex. (senars '(3 1 8 7 4 10)) → (3 1 7)

```
(defun senars (l)
  (cond ((null l) nil)
        ((oddp (car l)) (cons (car l) (senars (cdr l))))
        (t (senars (cdr l)))))

(defun parell (n)
  (cond ((= (* 2 (dividir n 2)) n) t)
        (t nil)))
```

8. Calcular les funcions màxim i mínim d'una llista de números

Ex. (maxim '(4 1 3 8 5)) → 8
(minim '(4 1 3 8 5)) → 1

```
(defun maxim (l)
  (cond ((null (cdr l)) (car l))
        ((>= (car l) (maxim (cdr l))) (car l))
        (t (maxim (cdr l)))))

(defun minim (l)
  (cond ((null (cdr l)) (car l))
        ((< (car l) (minim (cdr l))) (car l))
        (t (minim (cdr l)))))
```

9. Escriure una funció per ordenar una llista de números amb el mètode de selecció directa (trobar el mínim a cada passa i posar-ho al principi)

Ex. (ordena '(4 1 3 8 5)) → (1 3 4 5 8)

```
(defun borra (x l)
  (cond ((null l) nil)
        ((equal x (car l)) (cdr l))
        (t (cons (car l) (borra x (cdr l))))))

(defun ordena (l)
  (cond ((null l) nil)
        (t (cons (minim l)
                 (ordena (borra (minim l) l))))))
```

10. Escriure la funció invertir que donada una llista la gira al revés (sense utilitzar reverse).
(invertir '(a e i o u)) → (u o i e a)

```
(defun invertir (l)
   (cond ((null l) nil)
         (t (snoc (car l) (invertir (cdr l))))))
on

(defun snoc (x l)
   (cond ((null l) (list x))
         (t (cons (car l) (snoc x (cdr l))))))
```

11. Escriure una funció per esborrar l'enèsim element d'una llista.
(esborrar 3 '(a e i o u)) → (a e o u)

```
(defun esborrar (n l)
   (cond ((= n 1) (cdr l))
         (t (cons (car l)
            (esborrar (- n 1) (cdr l))))))
```

12. Escriure una funció per insertar un element a la posició enèsima d'una llista.
(inserta 'aqui 3 '(i jo que faig)) → (i jo aqui que faig)

```
(defun inserta (que on l)
   (cond ((= on 1) (cons que l))
         (t (cons (car l)
                  (inserta que (- on 1) (cdr l))))))
```

13. Escriure una funció per canviar l'enèsim element d'una llista per l'element donat.
(canviar 3 '(a e i o u) 'mig) → (a e mig o u)

```
(defun canviar (on l per)
```

```
   (cond ((= on 1) (cons per (cdr l)))
         (t (cons (car l)
            (canviar (- on 1) (cdr l) per)))))
```

14. Escriure una funció definida recursivament que compti el número de vegades que una expressió apareix a una llista.
Ex. (vegades 'a '(1 2 a 3 (a b) a)) → 2
(vegades 'x '(s t (x) 3)) → 0

```
(defun vegades (x l)
  (cond ((null l) 0)
        ((equal x (car l)) (+ 1 (vegades x (cdr l))))
        (t (vegades x (cdr l)))))
```

15. Definir una funció "contingut" que ens digui si una expressió és igual a una altra o si hi està continguda.

Ex. (contingut 'a 'a) → T
(contingut '(x) '(a (x (a b)) y)) → nil
(contingut '(a b) '(a (x (a b)) y)) → T

```
(defun contingut (x l)
  (cond ((equal x l) t)
        ((atom l) nil)
        (t (or (contingut x (car l))
               (contingut x (cdr l))))))
```

16. Escriure la funció "longituds" que transforma una llista canviant cada element per una parella formada per l'element original i la seva longitud.
Ex. (longituds '((a b) () (d))) → ((2 (a b)) (0 nil) (1 (d)))

```
(defun longituds (l)
  (cond ((null l) nil)
        (t (cons (list (length (car l))
                 (car l)) (longituds (cdr l))))))
```

17. Escriure una funció que compti el número total d'àtoms que hi ha dins una llista
Ex. (atoms '(a (b (c d) e) f g (h i))) → 9

```
(defun atoms (l)
  (cond ((null l) 0)
        ((listp (car l)) (+ (atoms (car l))
                            (atoms (cdr l))))
        (t (+ 1 (atoms (cdr l))))))
```