

ARIEL VILLEDA

TEORÍA COMPUTACIONAL  
FUNCIONES RECURSIVAS PRIMITIVAS

El proyecto se realizó utilizando el lenguaje de programación funcional **LISP** y el compilador de éste mismo llamado 'Steel Bank Common Lisp' (**SBCL**, por sus siglas en inglés).

Es importante mencionar que las funciones se implementaron para funcionar correctamente solo con Números Naturales.

Para realizar las implementaciones de las funciones solicitadas, adicionalmente se implementó la función inicial **sucesor**, y las funciones **monus** y **predecesor** de la siguiente manera:

**SUCESOR:**

|   |  |
|---|--|
| <pre>(defun sucesor(n)   (if (&lt; n 0)       0       (+ n 1)   ) )</pre> | La función retorna el número natural siguiente, si el número del que se desea obtener el sucesor es menor a cero, se retorna cero. |
|---|--|

**PREDECESOR:**

|  |   |
|--|---|
| <pre>(defun predecesor(n)   (if (&lt; n 1)       0       (- n 1)   ) )</pre> | La función retorna el número natural anterior, si el número del que se desea obtener el predecesor es menor a uno, se retorna cero. |
|--|---|

**MONUS:**

|  |  |
|--|--|
| <pre>(defun monus(n r)   (if (= r 0)       n       (predecesor(monus n (- r 1)))   ) )</pre> | Función recursiva primitiva con condición de parada cuando el sustraendo (r) es cero. Si el minuendo es menor que el sustraendo, retorna cero. |
|--|--|

A continuación, se hace una pequeña descripción de la implementación de las funciones recursivas parciales **suma**, **mult**, **expo**, **equ**, **coc**, **div**, **facto** y **fibo**; las funciones se realizan utilizando las funciones de menor nivel, por ejemplo, en **mult** se utiliza la función **suma**, y en **expo** se utiliza **mult**. Se adjunta una ejecución de esta.

## SUMA:

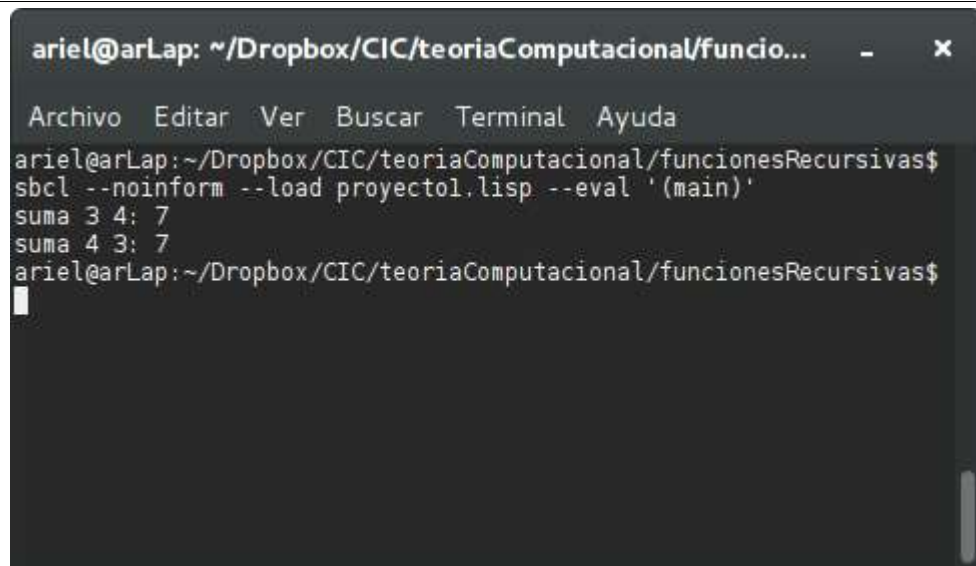
```
(defun suma(n s)
  (if (= s 0)
      n
      (sucesor(suma n (- s 1))))
  )
)
```

Función recursiva primitiva con condición de parada cuando el sumando (s) es cero.

### Ejemplo de ejecución

```
(format t "suma 3 4: ~d" (suma 3 4))(terpri)
```

```
(format t "suma 4 3: ~d" (suma 4 3))(terpri)
```



The screenshot shows a terminal window titled "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...". The terminal contains the following text:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
sbcl --noinform --load proyecto1.lisp --eval '(main)'
suma 3 4: 7
suma 4 3: 7
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
```

## MULT:

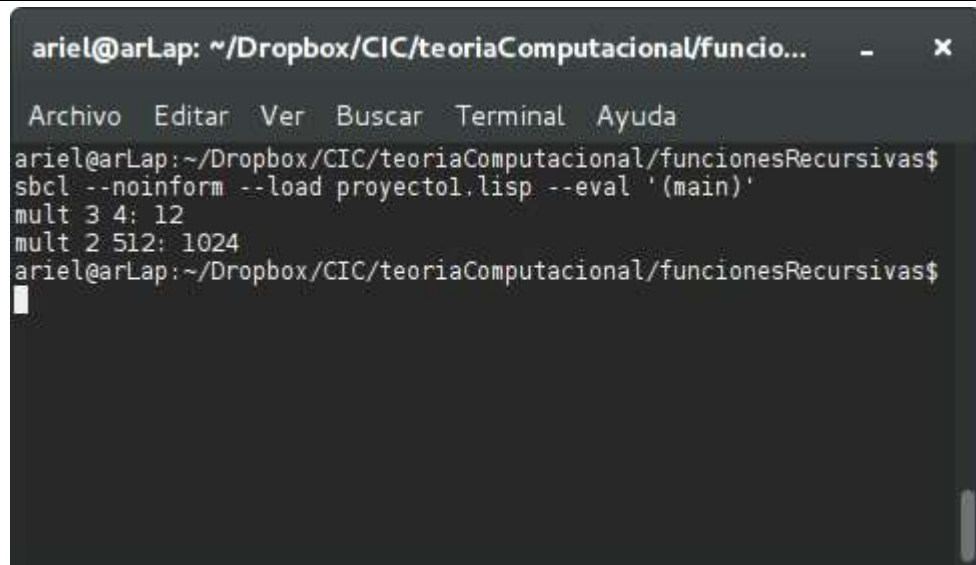
```
(defun mult(n m)
  (if (= m 0)
      0
      (suma n (mult n (- m 1)) )
  )
)
```

Función recursiva primitiva con condición de parada cuando se multiplica por cero.

### Ejemplo de ejecución

```
(format t "mult 3 4: ~d" (mult 3 4))(terpri)
```

```
(format t "mult 2 512: ~d" (mult 2 512))(terpri)
```



The screenshot shows a terminal window titled "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal content shows the user running the command "sbcl --noinform --load proyecto1.lisp --eval '(main)'" which results in the output "mult 3 4: 12" and "mult 2 512: 1024". The prompt "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas\$" is visible at the bottom.

```
ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
sbcl --noinform --load proyecto1.lisp --eval '(main)'
mult 3 4: 12
mult 2 512: 1024
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
```

## EXPO:

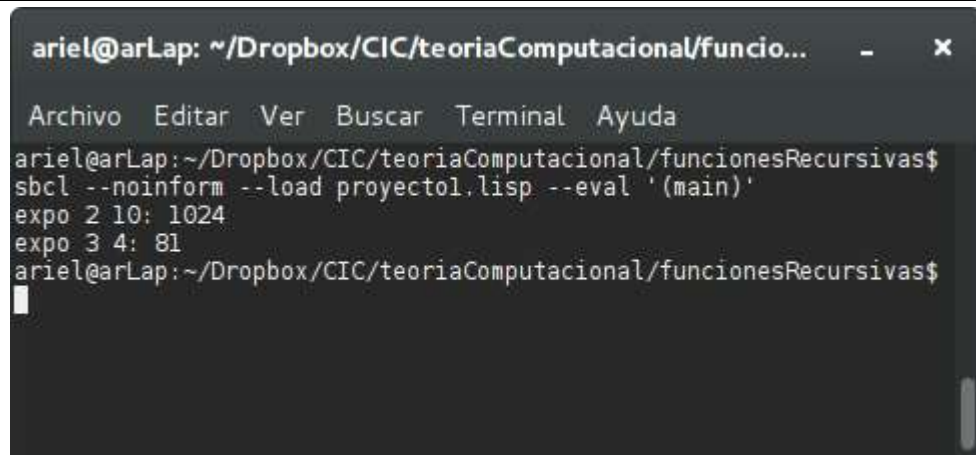
```
(defun expo(n e)
  (if (= e 0)
      1
      (mult n (expo n (- e 1)))))
)
```

Función recursiva primitiva con condición de parada cuando el exponencial disminuye a cero, en ese caso retorna 1.

### Ejemplo de ejecución

```
(format t "expo 2 10: ~d" (expo 2 10))(terpri)
```

```
(format t "expo 3 4: ~d" (expo 3 4))(terpri)
```



The screenshot shows a terminal window titled "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...". The terminal has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The prompt is "ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas\$". The user enters the command "sbcl --noinform --load proyecto1.lisp --eval '(main)'". The terminal outputs "expo 2 10: 1024" and "expo 3 4: 81". The prompt returns to "ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas\$".

```
ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio... - x
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
sbcl --noinform --load proyecto1.lisp --eval '(main)'
expo 2 10: 1024
expo 3 4: 81
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
```

## EQU:

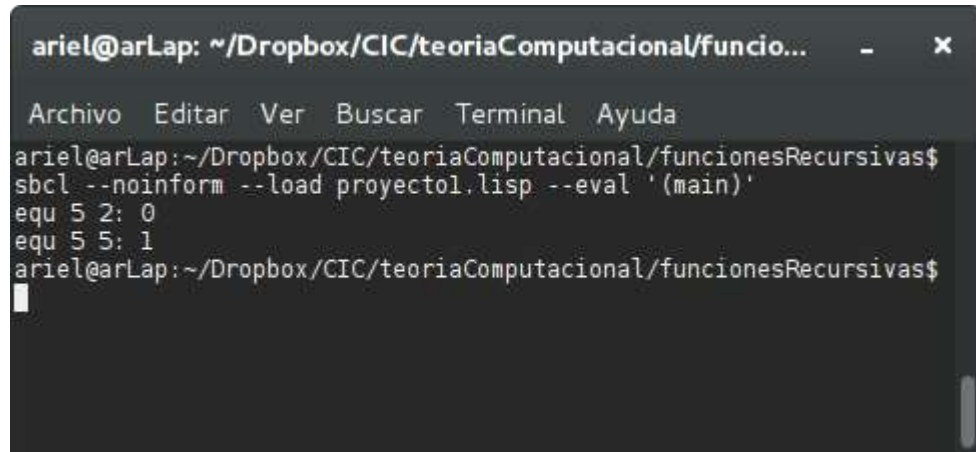
```
(defun equ(x y)
  (monus 1 (suma (monus y x) (monus x y))))
)
```

Función para comparación de valores, retorna 1 si los valores de n y c son los mismos. Retorna 0 en caso contrario.

### Ejemplo de ejecución

```
(format t "equ 5 2: ~d" (equ 5 2))(terpri)
```

```
(format t "equ 5 5: ~d" (equ 5 5))(terpri)
```



```
ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio... - x
Archivo Editar Ver Buscar Terminal Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
sbcl --noinform --load proyectol.lisp --eval '(main)'
equ 5 2: 0
equ 5 5: 1
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
```

## COC:

```
(defun coc(n c)
  (if (= n 0)
      0
      (suma (coc (- n 1) c) (equ n (suma (mult (coc (- n 1) c) c) c) ) )
  )
)
```

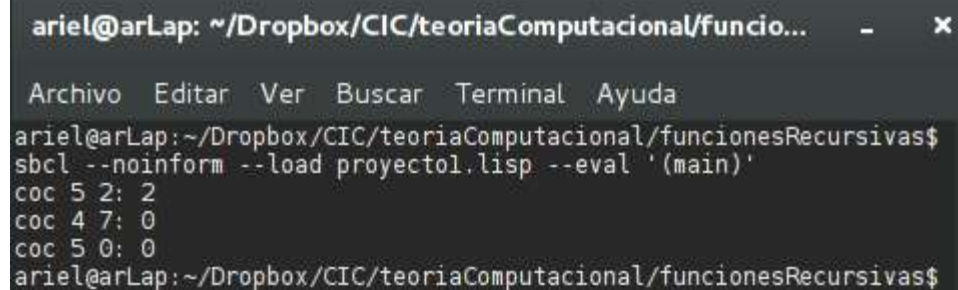
Retorna el cociente entero de dividir dos números. Si la división es entre cero o el dividendo es menor al divisor, retorna cero.

### Ejemplo de ejecución

```
(format t "coc 5 2: ~d" (coc 5 2))(terpri)
```

```
(format t "coc 4 7: ~d" (coc 4 7))(terpri)
```

```
(format t "coc 5 0: ~d" (coc 5 0))(terpri)
```



The screenshot shows a terminal window titled 'ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...'. The terminal contains the following text:

```
Archivo  Editor  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
sbcl --noinform --load proyectol.lisp --eval '(main)'
coc 5 2: 2
coc 4 7: 0
coc 5 0: 0
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
```

## DIV:

```
(defvar mint 0 )
(defun div(n d)
  (setq mint 0 )
  (if (> d 0)
    (loop
      (if (= (monus (+ n 1) (suma (mult mint d) d)) 0)
        (return mint) ; retornando minimizacion
      )
      (setq mint (+ mint 1))
    )
    'undefined ; retornando undefined
  )
)
```

Retorna el cociente entero de dividir dos números. Si en la división el dividendo es menor al divisor, retorna cero. Si es una división entre cero, retorna 'UNDEFINED' debido a la aproximación a minimización que utiliza ésta función.

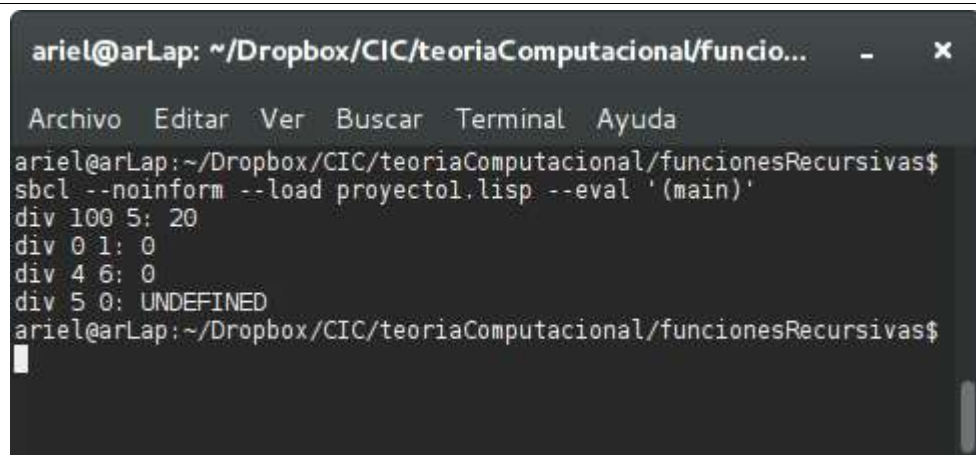
### Ejemplo de ejecución

```
(format t "div 100 5: ~d" (div 100 5) )(terpri)
```

```
(format t "div 0 1: ~d" (div 0 1) )(terpri)
```

```
(format t "div 4 6: ~d" (div 4 6) )(terpri)
```

```
(format t "div 5 0: ~d" (div 5 0) )(terpri)
```



The screenshot shows a terminal window titled "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...". The terminal contains the following text:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
sbcl --noinform --load proyectol.lisp --eval '(main)'
div 100 5: 20
div 0 1: 0
div 4 6: 0
div 5 0: UNDEFINED
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRekursivas$
```

## FACTO:

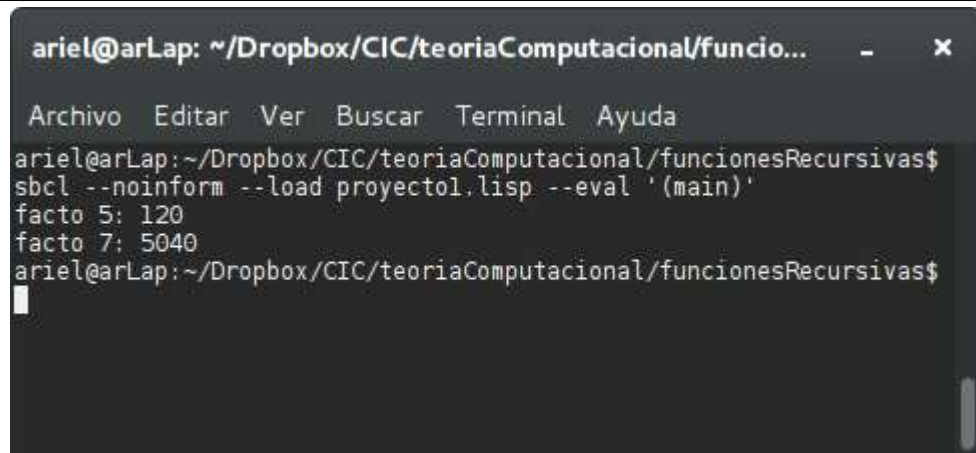
```
(defun facto(n)
  (if (< n 1)
      1
      (mult n (facto (- n 1)) )
  )
)
```

Función recursiva para la operación factorial de un número Natural. Caso base, el factorial de cero retorna 1. Hace uso también de la función recursiva **mult**.

### Ejemplo de ejecución

```
(format t "facto 5: ~d" (facto 5) )(terpri)
```

```
(format t "facto 7: ~d" (facto 7) )(terpri)
```



The screenshot shows a terminal window titled "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal content shows the user running the command "sbcl --noinform --load proyecto1.lisp --eval '(main)'" which results in the output "facto 5: 120" and "facto 7: 5040". The prompt "ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcionesRecursivas\$" is visible at the bottom.



## FIBO:

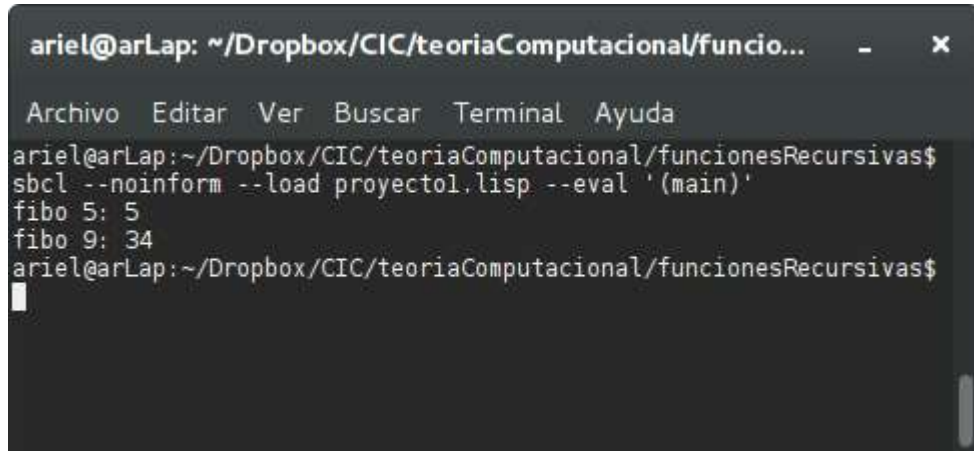
```
(defun fibo(n)
  (if (> n 1)
      (suma (fibo(- n 1)) (fibo (- n 2))) )
      (if (= n 1)
          1
          0
          )
      )
  )
```

Función recursiva para la sucesión de Fibonacci, cada recursividad necesita satisfacer las dos anteriores recursividades para resolverse, por lo tanto, como casos base tenemos dos valores, cuando  $n=0$  y cuando  $n=1$ .

### Ejemplo de ejecución

```
(format t "fibo 5: ~d" (fibo 5))(terpri)
```

```
(format t "fibo 9: ~d" (fibo 9))(terpri)
```

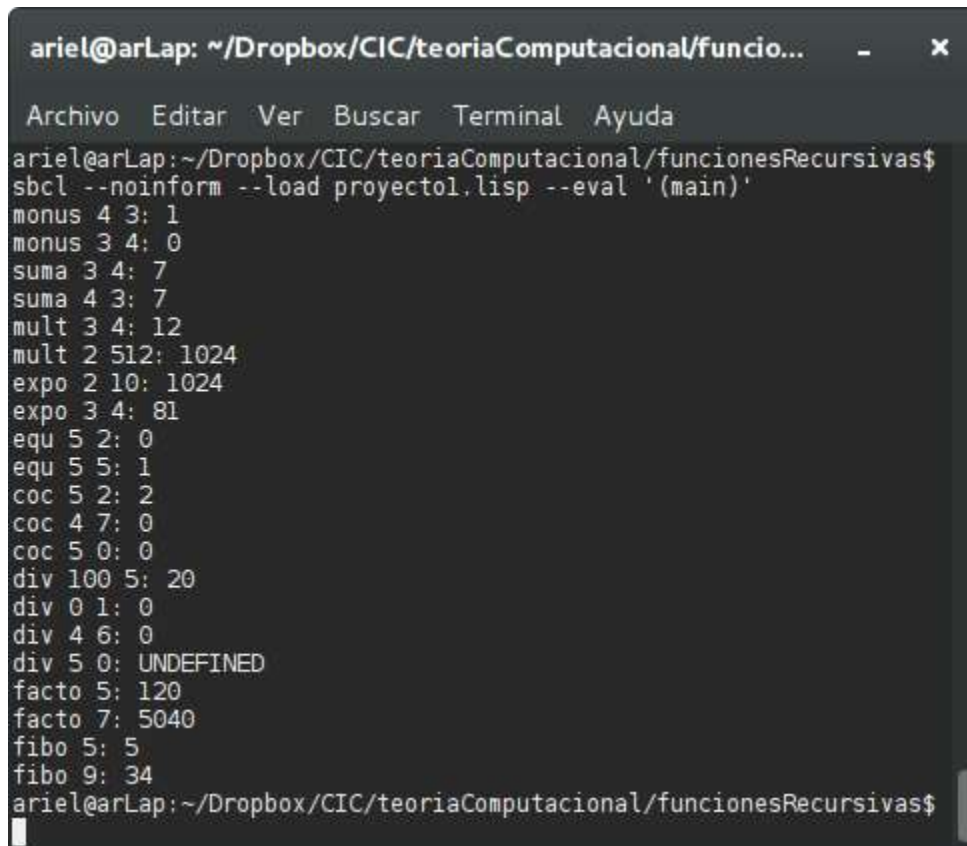


```
ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio...
Archivo Editar Ver Buscar Terminal Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
sbcl --noinform --load proyectol.lisp --eval '(main)'
fibo 5: 5
fibo 9: 34
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
```

El archivo adjunto **proyecto1.lisp** contiene todas las implementaciones de las funciones antes descritas, y muestras todos los ejemplos aquí mencionados con solo evaluar la función **main** del archivo. Si utiliza el compilador **SBCL** puede ejecutar el siguiente comando dentro de la carpeta contenedora del archivo **lisp** para poder evaluar la función principal antes mencionada:

***sbcl --noinform --load proyecto1.lisp --eval '(main)'***

Dando como resultado en terminal algo parecido a lo siguiente:



```
ariel@arLap: ~/Dropbox/CIC/teoriaComputacional/funcio... - x
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
sbcl --noinform --load proyecto1.lisp --eval '(main)'
monus 4 3: 1
monus 3 4: 0
suma 3 4: 7
suma 4 3: 7
mult 3 4: 12
mult 2 512: 1024
expo 2 10: 1024
expo 3 4: 81
equ 5 2: 0
equ 5 5: 1
coc 5 2: 2
coc 4 7: 0
coc 5 0: 0
div 100 5: 20
div 0 1: 0
div 4 6: 0
div 5 0: UNDEFINED
facto 5: 120
facto 7: 5040
fibo 5: 5
fibo 9: 34
ariel@arLap:~/Dropbox/CIC/teoriaComputacional/funcionesRecurtivas$
```