

Paradigmas de Programación

Práctica 5

La conjetura de Collatz

Ejercicios:

1. (Ejercicio opcional) Considere la función f de $\mathbb{N} \rightarrow \mathbb{N}$, que podría aproximarse en OCaml con la siguiente definición para `f: int -> int`

```
let f n = if n mod 2 = 0 then n / 2 else 3 * n + 1
```

Según la Conjetura de Collatz¹, si partimos de cualquier número positivo y vamos aplicando repetidamente esta función, seguiremos un camino que llegará inexorablemente al 1. Por ejemplo, partiendo del 13, tendríamos el siguiente camino:

13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Realice las siguientes tareas en un fichero de texto `collatz.ml`:

- Llamaremos “órbita” de un número al camino que se sigue de esta manera desde ese número hasta el 1 (ambos incluidos). Por ejemplo, la que hemos escrito más arriba es la órbita del 13.

Defina (de modo recursivo) una función `orbit: int -> string` tal que, cuando se aplique esta función a cualquier $n > 0$ se devuelva un *string* con la órbita de n , siguiendo el formato del ejemplo (los distintos valores por los que pasa la órbita deben estar separados por una coma y un espacio; no deben aparecer espacios ni al principio ni al final del *string*).

```
# orbit;;
- : int -> string = <fun>

# orbit 13;;
- : string = "13, 40, 20, 10, 5, 16, 8, 4, 2, 1"

# orbit 1;;
- : string = "1"
```

- Defina (de modo recursivo) una función `length: int -> int`, tal que, para cualquier $n > 0$, `length n` sea el número de pasos necesarios (en la órbita de n) para llegar hasta el 1. Así, por ejemplo, `length 13` debe ser 9.

```
# length;;
- : int -> int = <fun>

# length 13;;
- : int = 9

# length 27;;
- : int = 111
```

¹https://es.wikipedia.org/wiki/Conjetura_de_Collatz

- Defina (de modo recursivo) una función `top: int -> int`, tal que, para cada $n > 0$, `top n` sea el valor más alto alcanzado en la órbita de n . Así, por ejemplo, `top 13` debe ser 40.

```
# top;;
- : int -> int = <fun>
```

```
# top 13;;
- : int = 40
```

```
# top 27;;
- : int = 9232
```

- Defina directamente (de modo recursivo, y sin usar las funciones `length` y `top`) una función `length'n'top: int -> int * int`, tal que, para cada entero n , devuelva un par de enteros indicando la longitud de su órbita y su altura máxima. Así, por ejemplo, `length'n'top 13` debería ser el par (9, 40). Se trata de que al aplicar esta definición “no se recorra dos veces la órbita en cuestión”.

```
# length'n'top;;
- : int -> int * int = <fun>
```

```
# length'n'top 13;;
- : int * int = (9, 40)
```

```
# length'n'top 27;;
- : int * int = (111, 9232)
```

- Defina (usando recursividad) una función `longest_in: int -> int -> int * int` tal que `longest_in m n` devuelva el menor valor del intervalo $[m, n]$ cuya órbita tenga longitud maximal en ese intervalo, acompañado de dicha logitud maximal. De esta forma, por ejemplo, `longest_in 86 87` debería ser el par (86, 30).

```
# longest_in;;
- : int -> int -> int * int = <fun>
```

```
# longest_in 1 1000;;
- : int * int = (871, 178)
```

- Defina (usando recursividad) una función `highest_in: int -> int -> int * int` tal que `highest_in m n` devuelva el menor valor del intervalo $[m, n]$ cuya órbita tenga altura maximal en ese intervalo, acompañado de dicha altura maximal. De esta forma, por ejemplo, `highest_in 86 87` debería ser el par (87, 592).

```
# highest_in;;
- : int -> int -> int * int = <fun>
```

```
# highest_in 1 1000;;
- : int * int = (703, 250504)
```

El fichero `collatz.ml` es el único que hay que enviar cuando se solicite la entrega de esta práctica, y debe compilar sin errores con la orden `ocamlc -c collatz.mli collatz.ml`.