La conjetura de Collatz

Considere la función f de N -> 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 Collatz**[1], si partimos de cualquier número positivo y aplicamos repetidamente esta función seguiremos un camino que llegará inexorablemente al 1.

La función definida a continuación puede usarse para comprobar si un número en particular verifica esta conjetura, en el sentido de que si, al aplicar esta función a dicho número, el cálculo termina, la conjetura se corrobora en ese número.

```
let rec verify n =
    n = 1 || verify (f n)
```

Así, el siguiente ejemplo comprueba la veracidad de la conjetura en el número 871.

```
# verify 871;;
- : bool = true
```

En un archivo con nombre ***collatz.ml,*** defina una función ***verify_to : int -> bool***, de modo que *verify_to n* devuelva *true* después de comprobar la conjetura en todos los naturales hasta el *n* (incluido). No debería llevar mucho tiempo comprobar con esta función que la conjetura se cumple para, digamos, el primer millón de números naturales[2].

```
# verify_to;;
- : int -> bool = <fun>
# verify_to 1_000_000;;
- : bool = true
```

Llamaremos "órbita" de un número al camino que se obtiene al aplicar repetidamente, desde ese número, la función *f* hasta llegar al *1*. Así, por ejemplo, la órbita del *13* sería:

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

---

[1] Una conjetura es una afirmación que se supone cierta (basándose sobre todo en la observación y en que no se conocen contra-ejemplos), pero para la que no se dispone de una demostración formal. Lothar Collatz enunció su famosa conjetura en 1937, dos años después de doctorarse, y aun no se ha podido probar su veracidad o falsedad.

[2] La conjetura de Collatz ha sido comprobada por ordenador para todos los números hasta, al menos, 2^{68}; lo cual no implica, en absoluto, que no puedan existir contra-ejemplos mayores.

Añada al mismo archivo la definición en OCaml (de modo recursivo) de una función ***orbit : int -> string*** tal que, cuando se aplique esta función a cualquier *n > 0*, devuelva un string con la representación de su órbita siguiendo exactamente el formato de los siguientes ejemplos (los valores deben ir separados por una coma y un espacio en blanco; no hay coma ni espacio en blanco después del último valor):

```
# orbit 13;;
- : string = "13, 40, 20, 10, 5, 16, 8, 4, 2, 1"
# orbit;;
- : int -> string = <fun>
# orbit 13;;
- : string = "13, 40, 20, 10, 5, 16, 8, 4, 2, 1"
# orbit 1;;
- : string = "1"
```

Añada al mismo archivo la definición (recursiva) de una función ***length: int -> int***, tal que, para cualquier *n > 0*, *length n* sea el número de valores que contiene la órbita de *n*. Así, por ejemplo, *length 13* debe ser *10*.

```
# length;;
- : int -> int = <fun>
# length 13;;
- : int = 10
# length 27;;
- : int = 112
#
```

Añada, por último, al mismo archivo, la definición (recursiva) de 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
#
```

Opcional: Defina también directamente (de modo recursivo, 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 *(10, 40)*. Se trata de que al aplicar esta definición "no se recorra dos veces la órbita en cuestión". (Esta definición debe incorporarse a un archivo de nombre ***collatz_plus.ml***)

```
# length'n'top;;
- : int -> int * int = <fun>
# length'n'top 13;;
- : int * int = (10, 40)
# length'n'top 27;;
- : int * int = (112, 9232)
#
```

Puede comprobar que tiene definidas todas las funciones solicitadas con los tipos adecuados utilizando los archivos de interfaz ***collatz.mli*** y ***collatz_plus.mli***