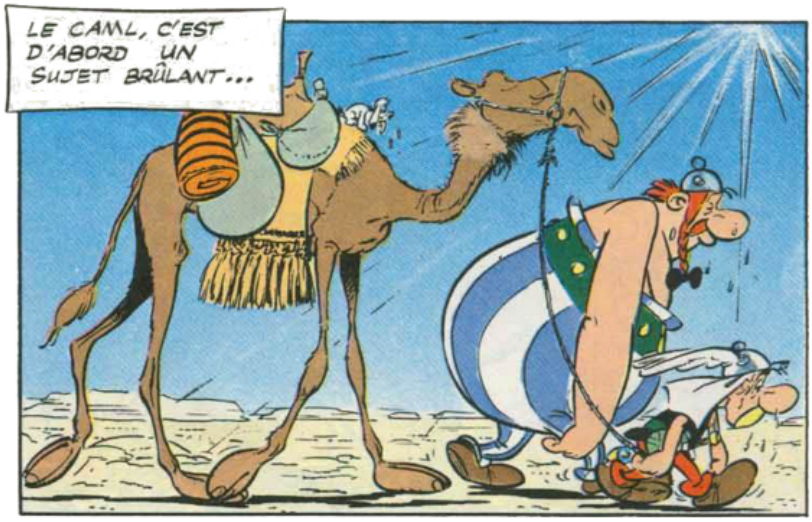
Algorithms

Undergraduate S1

EPITA

Intensive CAML Session Exercises

September 2025



1 Basics **1**

- Expressions and Definitions . . . 1
- Basic Data Types . . . 2

2 Functions **3**

- One Parameter . . . 3
- Several Parameters . . . 4

3 Case analysis **5**

- Conditional . . . 5
- Pattern matching . . . 7
- Pattern matching, anonymous functions and cartesian product . . . 9

4 Recursivity **11**

Unless otherwise stated, the environment is assumed to be **empty** at the beginning of each exercise.

1 Basics

Expressions and Definitions

Exercise 1.1 (Evaluations)

What will the successive CAML evaluation results of the following phrases be?

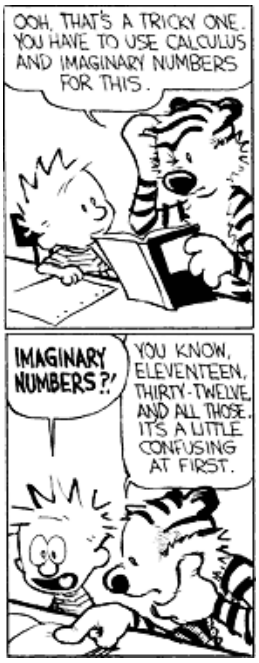
```
# let a = 11 + 7 * 3 ;;

# let b = 3 in 6 + b ;;

# let a = 3 in let c = 2 + a ;;

# let c = let a = 3 in 2 + a ;;

# let c = a * b ;;


# let a = 2 in b = 5 in a * b ;;

# let a = 5 in let b = 2 * a in a * b ;;

# let a = 5 and b = 2 * a in a * b ;;
```


```
# (let x = 2 in x * (x - 1)) + (let x = 3 in x * (x + 1)) ;;

# let x = 2 and y = 5 in
  let y = x and z = y in
  y + z ;;

# let x = 2 and y = 5 in
    let y = x in
      let z = y in
        y + z ;;

# let a=let a=let a=1 in 2*a and b=let b=2 in 2*b in let a=a+b and b=a-b in a*a-b*b;;
```

Basic Data Types

Exercise 1.2 (Types and Values)

What are the types of the following expressions and what do they evaluate to?




```
# 1. +. 2. ;;
# 1 + 1.5 ;;
# 1 +. 1.5 ;;

# 1 / 2 ;;
# 5 / 2 = 4 / 2 ;;
# 5 mod 2 ;;
# 4 mod 2 ;;

# 10-2 * 2 ;;
# 10 * 24 mod 10 ;;
# 24 mod 10 * 10 ;;

# "A string" ;;
# "" ;;
# 'a' ;;
# '' ;;
# "Hello Word" ^ '!' ;;

# 1. < 3 ;;
# 1 < 2 < 3 ;;
# "how do string comparisons work?" > "how do strings work?" ;;
# 'Z' < 'a' ;;
# let f x = x + 1 and g x = x + 1 in f = g ;;

# 1 < 2 && 3. > 5. ;;
# false || (1/0 = 0) ;;
# true || (1/0 = 0) ;;
# true && (1/0 = 0) ;;
# false && (1/0 = 0) ;;
# false && (1.5 > 1) ;;
```

Exercise 1.3 (Logic)

Let a, b and c be three boolean values.

1. Simplify the following CAML expressions:

   - a `&&` `true`
   - a `&&` `false`
   - a `&&` `not` a
   - a `||` `true`
   - a `||` `false`
   - a `||` `not` a
   - (`(`a `&&` `not` b`)` `||` `(``not`` `$a$ `&&` $b$`)`
   - (`(`a `&&` b`)` `||` `(``not`` `$a$ `&&` `not` $b$`)`

2. Replace the ?? by the suitable operator.

   - `not (`a `&&` b`)` is equivalent to `not` a `??` `not` b
   - `not (`a `||` b`)` is equivalent to `not` a `??` `not` b

3. What expression(s) is (are) equivalent to the below expression?

```
(a && b) || (a && c)
```

   - `a && (b || a) && c`
   - `a && (b || c)`
   - `(a && b) || c`

2 Functions

One Parameter

Exercise 2.1 (Function types and applications)

The following functions are in the environment:

- `sqrt : float -> float` (square root)
- `float_of_int : int -> float`
- `int_of_float : float -> int`

Evaluate the following phrases and correct any possible mistakes:

```
# let f x = x * x ;;
# f -4 ;;
# f 2 + 3 ;;
# f (2 + 3) ;;

# let g x =
      let f x = x + 1 in
        f x*2 ;;
# g 5 ;;

# let f1 x = int_of_float x / 2 ;;
# let f2 x = sqrt float_of_int f1 x ;;

# let f3 x = x mod 2 = 0 in let f4 x = f3 x+1 in f4 5 ;;

# let x = 12 and f x = 2 * x ;;
# let a = 3 and f x = x + a ;;

# let g1 x = true && (x/0 = 0) ;;
# g1 0 ;;
# let g2 x = false && (1.5 > x+1) ;;
# g2 0 ;;
# let g3 x = let a = 12 in x * x ;;
# g3 30 ;;
```

Exercise 2.2 (Power)

Write the function `power30 : int -> int` that takes x as parameter and computes x^{30}: Multiplication is the only allowed operator, the fewer there are, the better it is!

Exercise 2.3 (Take a Chance)

Write the following CAML functions:

mirror: takes a positive integer n with **two non-zero digits**[1] as parameter and returns its "mirror".

abba: Use `mirror` to write a function which for any number with two non-zero digits ab computes $abba$.

stammer: Use the two previous functions to write the function which for any number of the form ab returns $baababba$.

Application examples:

```
# mirror 15 ;;
 - : int = 51
# abba 42 ;;
 - : int = 4224
# stammer 18 ;;
 - : int = 81181881
```

[1] For the moment, we will not mind results when the parameter is not a non-zero [illegible] integer.

Several Parameters

Exercise 2.4 (Function Applications)

Evaluate the following phrases and correct any possible mistakes.

```
# let sum a b c d = a + b + c + d ;;
# sum 2 0 2 4 ;;

# let build a b c d = a * 1000 + b * 100 + c * 10 + d ;;
# build 2 0 2 9 ;;

# let f x y = x mod y ;;
# let g x y z = x + y = z ;;
# f (2, 5) ;;
# let a = 20 and b = 3 in f (a b) ;;
# g -4 -5 -9 ;;
# f (f 25 7) (f 4 3) ;;
# g (f 5 2 * 2) (5 - f 13 4) (f 20 7) ;;
# f (g 5 2 7) (4 * 12) ;;
```

Exercise 2.5 (Function Types)

`float_of_int : int -> float` and `int_of_float : float -> int` are in the environment.
Give the types of the following functions:

```
# let divmod x y = (x / y, x mod y) ;;

# let test a b c = (float_of_int a *. b) > 4. && c = 'A' ;;

# let chapi x y =
     let a = 3. *. x in
        let b = int_of_float a mod y in
              b = 0 ;;

# let chapo x y z =
     let g a =
         let b = float_of_int (a * x) in
             (b +. y) /. 2.
     in int_of_float (g z) ;;
```

Exercise 2.6 (Hours, Minutes, Seconds)

The parameters are assumed "valid" (minutes and seconds in $[0, 60[$, hours $\in \mathbb{N}$).

1. Write the function `sec_of_time` which converts a duration represented by three integers (hours, minutes, seconds) into seconds. Example: 23h 13m 59s = 83639s
2. Write the function `time_of_sec` which converts seconds in hours, minutes and seconds (which will be represented by the triple (h, mn, s)).
3. Use the two previous functions to write the function `add_times` which adds two durations (each represented by 3 integers). Example: 12h 22m 15s + 15h 13m 10s = 27h 35m 25s

Application examples:

```
# sec_of_time 23 13 59 ;;
- : int = 83639

# time_of_sec 83639 ;;
- : int * int * int = (23, 13, 59)

# add_times 12 22 15 15 13 10 ;;
- : int * int * int = (27, 35, 25)
```

3 Case analysis

Conditional

Exercise 3.1 (Evaluations)

Evaluate the following phrases. Whenever possible, write more simple equivalent functions.

```
# let test b = if b then true else 0 ;;

# let add a b =
    (if a > b then a - b else b - a)
    + (if a <= b then b / a else a / b) ;;
# add 10 5 ;;

# let f a b c =
    (if a > b then
        if b > c then a + b else c + a
     else
        if a > c then a + b else b + c) *
      (if a > b then
         if b > c then a - b else a - c
       else
         if a > c then a - b else b - c) ;;
# f 1 2 3 ;;

# let all_even a b c d =
      if a mod 2 = 0 then
        if b mod 2 = 0 then
          if c mod 2 = 0 then
            if d mod 2 = 0 then
              true
            else
              false
          else
            false
        else
          false
      else
        false ;;

# all_even 0 2 4 6 ;;
```

Exercise 3.2 (Conditional vs logical)

For each logical formula below, write a CAML boolean function using only conditional expressions (without boolean operators) while trying to simplify as much as possible.

1. $a \wedge b$ logical *and*
2. $a \vee b$ logical *or*
3. $\neg a \vee b$ logical *implication*: $a \rightarrow b$

Find for this two last formulas an equivalent non logical operator.

4. $(a \wedge \neg b) \vee (\neg a \wedge b)$ logical *exclusive or*
5. $(a \rightarrow b) \wedge (b \rightarrow a)$ logical *equivalence*: $a \equiv b$

Exercise 3.3 (Orders)

Define the following functions:

1. `max2` et `min2` which respectively return the maximum and the minimum values of two integers ;
2. `max3` et `min3` which respectively return the maximum and the minimum values of three integers **without** using the predefined / previous functions ;
3. `middle3` which returns the value of the middle among 3 integers (you can use previous functions) ;
4. `max4` et `min4` which respectively return the maximum and the minimum values of four integers: (use the previous functions!).

Exercise 3.4 (Sum of Squares)

Write the function `highest_square_sum` which for three given integers returns the sum of the squares of the two highest.

Use the functions defined at the exercise 3.3.

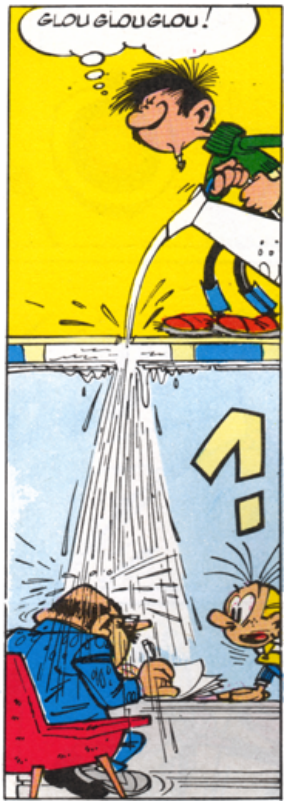
Application examples:

```
# highest_square_sum 3 2 1 ;;
- : int = 13

# highest_square_sum 1 (-5) 10 ;;
- : int = 101

# highest_square_sum (-1) (-2) (-3) ;;
- : int = 5
```

Pattern matching

Exercise 3.5 (Evaluations)

What will be the evaluation results of the following phrases? Correct any possible mistakes and try to eliminate the "warnings".




```
# let f1 x = match x with
    | 0 -> 0
    | y -> 1 / y ;;

# let switchonoff x = match x with
      "on" -> true
    | "off" -> false ;;

# let a = 42 and b = 666 in
    match a * b - 27972 with
      | 0            -> "zero"
      | x when x > 0 -> "positive"
      | _            -> "negative" ;;

# let square x = match x with
      x when x > 0  -> x * x
    | x when x <= 0 -> invalid_arg "x has to be positive" ;;
# square 5 ;;
# square (-5) ;;

# let parity n =
    if n >= 0 then
      match n with
          0 | 2 | 4 | 6 | 8 -> "even"
        | 1 | 3 | 5 | 7 | 9 -> "odd"
    else
      failwith "negative";;
  in
    parity 8 ;; (* test with (-10), 34 ... *)

# let grade a b =
    match (a + b) / 2 with
        x when x > 15 -> 'A'
      | x when x > 5 -> 'D'
      | 8 | 9 | 10    -> 'C'
      | x when x > 11 -> 'B'
      | _             -> 'E' ;;
# grade 12 14 ;;

# let h x y =
    match x with
        0 -> 0
      | y -> 1
      | _ -> -1 ;;
# h 5 5 ;;
# h 5 3 ;;
```

Exercise 3.6 (Simplifications)

Give the types of the following functions. What do they compute?
Write more simple equivalent functions!

1.
```
# let test a =
    let f n =
      if n < 0 then
        -1
      else
        1
    in
    match f a * a with
        0 | 1 | 2 | 3 | 4 -> true
      | n when n >= 10 -> false
      | _ -> true ;;
```

2.
```
# let f a b = match a with
      0             -> 0
    | x when x < 0  -> (match b with
                          0  -> 0
                        | _  -> -x + x/b*b)
    | x             -> (match b with
                        | 0          -> 0
                        | y when y<0 -> x - (x/(-y))*(-y)
                        | y          -> x - x/y*y) ;;
```

Exercise 3.7 (Postage)

The aim of this exercise is to write a fonction calculating the postage rate of a package under 3kg for shipments in France. Rates are given in the following table:

Weight up to	Economic	Standard	Express
500g	3,4 €	4,6 €	9,1 €
1000g	4,6 €	5,9 €	11 €
2000g	5,1 €	6,5 €	13,5 €
3000g	6,9 €	7,2 €	14,2 €

The 3 kinds of rates will be represented using the character strings `"eco"`, `"standard"` and `"express"`.

1. Write the function `rate_eco` which computes the postage rate for shipping a package in economic, based on the weight.
2. In the same way as the `rate_eco` function, write `rate_standard` and `rate_express`, both functions which compute the postage rate for shipping a package in standard and express.
3. Write the function `rate` which allows us to compute the postage rate for shipping a package knowing its weight and the rate code.
4. Can we solve the problem using fewer functions? Try to combine the rate functions `rate_eco`, `rate_standard` and `rate_express` in one.

```
# rate 750 "eco" ;;
 - : float = 4.6
# rate (-150) "standard" ;;
 Exception: Invalid_argument "weight has to be positive".
# rate 150 "drone" ;;
 Exception: Failure "unknown code".
# rate 3500 "express" ;;
 Exception: Failure "too heavy".
```

Pattern matching, anonymous functions and cartesian product

Exercise 3.8 (Evaluation)

What will the successive evaluations of the following phrases give?

```
# let a = let b = (0, true) in (b, "one") ;;
# let (x, y) = a in x ;;

# let (a, b) = (1, 2) ;;
# let (c, _) = (a, b) ;;

# let f c = let (x, y) = c in let z = (x+1, not y) in (x, z) ;;

# let g x y z = match (z, x) with
          (1, _) -> (1, x)
        | (y, true) -> (2 * y, x)
        | _ -> failwith "error" ;;
# g true "" 2 ;;

# let f x = match x with
          ((1, true), _)  -> 1
        | ((2, false), x) -> 3
        |  (x, "aa")      -> let (a, b) = x in a
        |  _              -> 2 ;;

# let f x y =
      let g x = (x + 1)/ 2
      in
      match (not x, g y) with
        (true, _) -> true
      | (_, 42) -> false
      | _ -> failwith "error" ;;

# let x = 5 in function a -> function b ->
  let y = 2 in float_of_int (a * x) > (float_of_int y +. b) ;;

# let test = function
      | (true, b) -> b
      | _         -> false ;;

# let f = function
       ((0, _), _) | (_, (0, _))       -> (0, false)
     | ((x, sx), (y, sy)) when sx = sy -> (x * y, false)
     | ((x, true), (y, sy))            -> (x * y, not sy)
     | ((x, _), (y, sy))               -> (x * y, sy) ;;

# let f = function
      "and" -> (function (v1, v2) -> v1 && v2)
    | "or" -> (function (v1, v2) -> v1 || v2)
    | "xor" -> (function (v1, v2) -> v1 <> v2)
    | "imply" -> (function (v1, v2) -> not v1 || v2)
    | _ -> failwith "unknown operator" ;;

# let h x y = match (x, y) with
        (x, x) -> true
      | _      -> false  ;;
```

Exercise 3.9 ("Anonymous Pattern Matching")

The functions to write must necessarily use pattern matching but without `match`!

1. Write a function that counts the number of 0 in an integer pair.

```
# count (0, 0) ;;
 - : int = 2
# count (0, 1) ;;
 - : int = 1
# count (4, 5) ;;
 - : int = 0
```

2. Write the following function: $\forall (m,n) \in \mathbb{Z} \times \mathbb{Z}, f(m,n) = \begin{cases} 0 & \text{if } m = 0 \\ 2m & \text{if } m \neq 0 \,\&\, n = 0 \\ m.n & \text{if } m \neq 0 \,\&\, n \neq 0 \end{cases}$

3. Let the CAML function `or3` be defined below. Find an equivalent function without using logical operators.

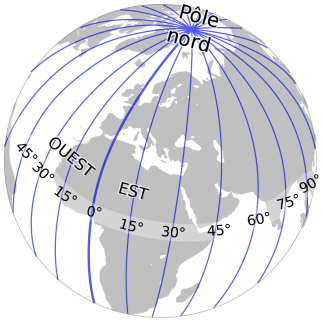
```
# let or3 (a, b, c) = a || b || c ;;
```

Exercise 3.10 (Jet lag)

We want to compute the solar time lag given in h, m, s between two places on earth, of which are known the longitude in °, ' and " West ('W') or East ('E') compared with the Prime meridian (which runs through Greenwich, England). For example, the point 3°12'15"E is 1h5mn31s in advance in comparison with the point 13°10'30"W.
Longitudes are given from 180°W to 0°E, and from 0°W to 180°E. The longitude of the change of date line (dashed line) is 180°.
To determine the solar time lag between two points, thereby avoiding the problems of date change, we will compute for each point its lag with Greenwich, then we will compute the difference between these two lags.

Clue : $360/24 = 15$
$\Rightarrow 15^o = 1h$

$\Rightarrow$ 3°12'15"E = 11535"
= 769s
= 0h 12 mn 49 s



Write a function `time_difference` which returns the time lag between two points of given longitudes.
Advice :
Start by breaking down the problem into sub-problems to determine the necessary functions.

Examples:

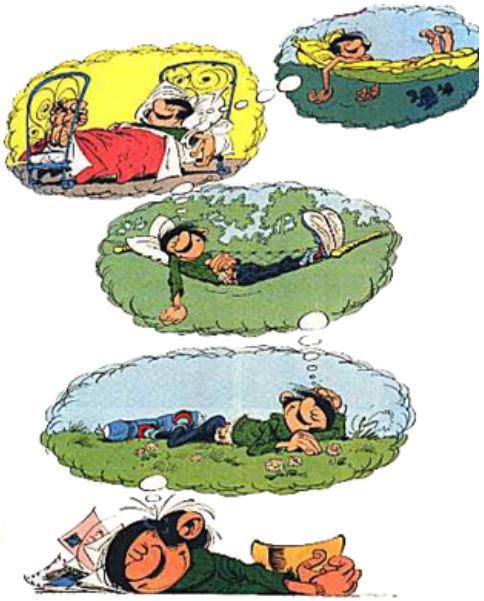
```
# time_difference (3, 12, 15, 'E') (0, 0, 0, 'W');;
 - : string = "earlier:0h 12mn 49s"

# time_difference (3, 12, 15, 'E') (13, 10, 30, 'W') ;;
 - : string = "earlier : 1h 5mn 31s"

# time_difference (0, 1, 4, 'W') (2, 0, 5, 'E') ;;
 - : string = "later: 0h 8mn 4s"
```

4 Recursivity

Exercise 4.1 (What is it?)

What do the following functions compute (consider all cases)?

```
# let rec f1 n = match n with
      0 -> 0
    | n -> n + f1 (n-1) ;;

# let rec f2 n d = match n with
      n when n < 0       -> invalid_arg "f2: n < 0"
    | 0                  -> 0
    | n when n mod d = 0 -> n + f2 (n-1) d
    | n                  -> f2 (n-1) d ;;

# let rec f3 d n =
    if n <= 1 then
      d
    else
      d * f3 (d - 1) (n - 1) ;;

# let rec f4 = function
      0 -> 0
    | n -> n / 2 + f4 (2*n) ;;

# let rec mystery x =
    if x < 10 then
      1
    else
      1 + mystery (x/10) ;;
```



What will be the results of the evaluations of the following phrases and what will be displayed?

```
# let rec print1 n =
    if n = 0 then
      ()
    else
      begin
        print_int n; print_char ' ';
        print1 (n-1)
      end ;;

# print1 5 ;;
```

```
# let rec print2 n =
    if n = 0 then
      ()
    else
      begin
        print2 (n-1);
        print_int n; print_char ' ';
      end ;;

# print2 5 ;;
```

```
# let rec g n d =
    if n = 0 then
      0
    else
      begin
        print_int n; print_char ' '; print_int d; print_newline();
        if n mod d = 0 then
          1 + g (n/10) d
        else
          g (n/10) (d-1);
      end ;;

# g 435567 7 ;;
```

Exercise 4.2 (Sequence)

Write a function which computes from a given $n \in \mathbb{N}$ the n^{th} element value of the sequence defined by $u_n = 4u_{n-1} + 2$ from $u_0 = 1$.

Exercise 4.3 (Geometric progression)

Write a recursive function which computes the n^{th} ($n \in \mathbb{N}$) element of a geometric progression from u_0 and of common ratio q.

Exercise 4.4 (Euclid)

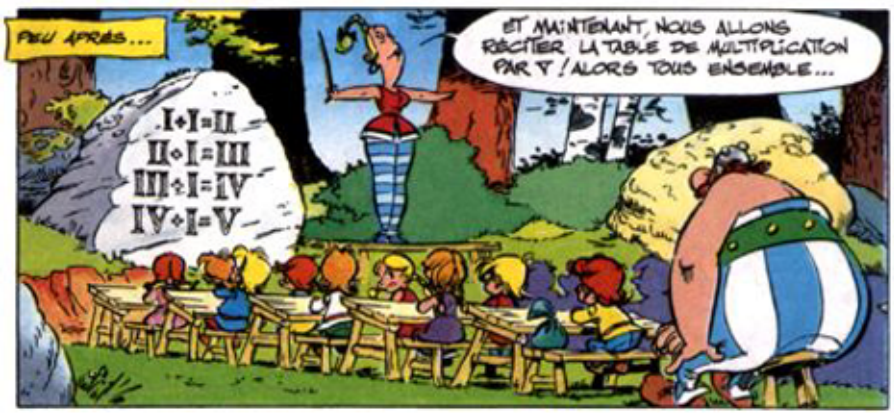
Write a function that computes the greatest common divisor (GCD) of two nonzero integers using the Euclidean algorithm, the principle of which is "reminded" below.

Euclidean algorithm :

> If a and b are two nonzero naturals with $a \geq b$, if r is the nonzero remainder of the division of a by b: $a = bq + r$ with $0 < r < b$, then the GCD of a and b is equal to the GCD of b and r.
> If a is divisible by b then the GCD of a and b is equal to b.

Exercise 4.5 (Addition)

Write a CAML function that computes $x + y$ ($(x, y) \in \mathbb{N}^2$) only adding 1.



Exercise 4.6 (Multiplication)

Write a function which computes $x \times y$ ($(x, y) \in \mathbb{N}^2$) using only the operators + and -.

How to handle the cases where $(x, y) \in \mathbb{Z}^2$?

Exercise 4.7 (Euclidean division)

Reminder: Given two naturals a and b, with $b \neq 0$, the Euclidian division gives two naturals the quotient q and the remainder r such that:

$$a = bq + r, r < b$$

1. Write a function that computes the quotient of the division of a by b with a and b two naturals. The function must use only additive operators (the allowed operators are + and -).
2. Write a function that computes the remainder of the integer division of a by b. The function must use only additive operators.
3. **Bonus:** Write a function that computes the quotient and the remainder (a pair) of the Euclidian division of two naturals (use a single recursive function).

Exercise 4.8 (Fibonacci)

Write a function which computes the n^{th} term ($n \in \mathbb{N}$) of the Fibonacci numbers defined below:

$$F(n) = \begin{cases} 1 & if\ n \leq 1 \\ F(n-1) + F(n-2) & otherwise \end{cases}$$

Exercise 4.9 (Mirror)

Write a function which "reverses" the digits of any natural number.

Two versions :

1. The result will be a character string containing the "mirror" number.
   Use the function `string_of_int : int -> string`.

```
# reverse 1234 ;;
- : string = "4321"
```

2. The result will be of integer type (neither using the previous function, nor character strings).

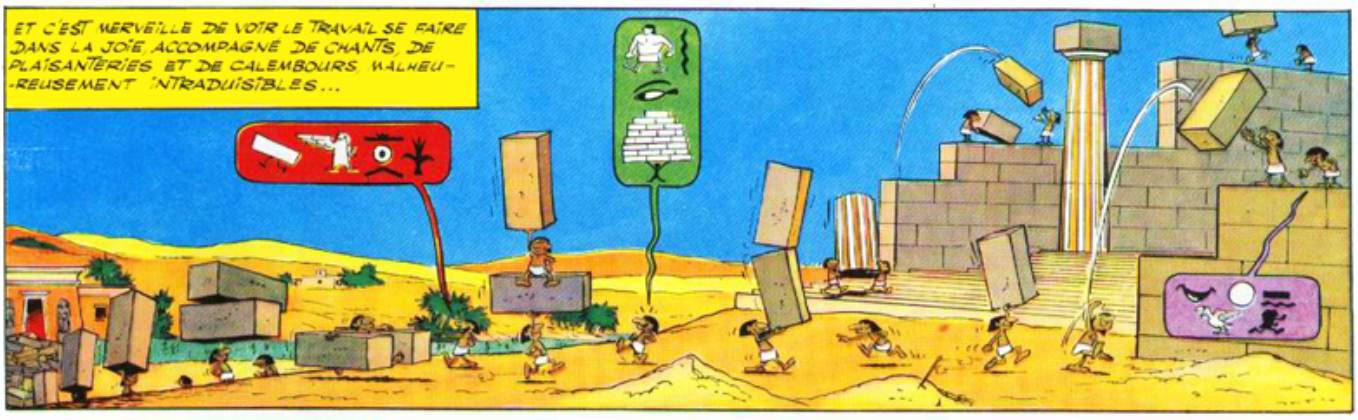
```
# reverse_int 1234 ;;
- : int = 4321
```

Exercise 4.10 (Egyptian multiplication)

Write a function that computes $x \times y$ ($(x, y) \in \mathbb{N}^2$) only using additions, multiplications by 2 and divisions by 2.

Clues :

- $10 * 13 = 2 * (5 * 13)$ because 10 is even.
- $11 * 13 = 2 * (5 * 13) + 13$ because 11 is odd.



Exercise 4.11 (Power)

Let $x \in \mathbb{R}$ and $n \in \mathbb{N}$.

1. Write a recursive function which computes x^n using only the "classical" method (by successive multiplications).
   What is the complexity order of this version?

2. Write a new function which computes x^n using the *exponentiation by squaring* principle: be inspired by the Egyptian multiplication principle (exercise 4.10) for reducing the number of multiplications.
   What is the complexity order of this version?

Exercise 4.12 (Prime number)

Write a function that determines whether an integer greater than 1 is a prime number.

Exercise 4.13 (Perfect)

A number is a perfect number if it is a integer greater than 1 that is equal to the sum of its proper positive divisors, which are its positive divisors excluding the number itself.
Example : $28 = 1 + 2 + 4 + 7 + 14$.

Write a function which determines if a natural number n given ($n \neq 1$) is perfect.

Bonus:
The function could display the divisors of n.

Display example:

```
Divisors:
1
2
4
7
14
28 is perfect
```

How to improve the function complexity?

© All pictures used to illustrate this document are property of their authors and publishers. Should those authors or publishers wish to have them removed, this will be done immediately upon their request.