# Lists

## 1 Simple traversals

### Exercise 1.1 (Product)

Write a function that calculates the product of all elements of an integer list.

### Exercise 1.2 (Count)

Write a function that counts the number of a given value in a list.

### Exercise 1.3 (Search)

Write a function that tests whether an element is present in a list.

### Exercise 1.4 (Check occurrences – *C1# 05/2021*)

Write the function `check_occ` *x n list* that checks if element *x* is in the list *list* at least *n* times.

```
# check_occ 1 1 [1; 4; 3; -10];;
- : bool = true
# check_occ 1 2 [1; 4; 3; -10];;
- : bool = false
# check_occ 1 2 [4; 1; 1; -10];;
- : bool = true
# check_occ "td" 1 ["caml"];;
- : bool = false
# check_occ "td" 3 ["td"; "caml"; "td"; "2021"; "td"; "Python"; "td"];;
- : bool = true
```

### Exercise 1.5 (n$^{th}$)

Write a function that calculates the $n^{th}$ element of a list. The function has to raise an exception `Invalid_argument` if $n$ is negative or zero, or an exception `Failure` if the list is too short.

### Exercise 1.6 (Maximum)

Write a function that returns the maximum value of a list.

### Exercise 1.7 (Bonus: Second)

Write a function that returns the second smallest element of a list if it exists.

*Application examples:*

```
    # second [1;3;4;2] ;;
    - : int = 2

    # second [3.5;8.2;9.5;4.0];;
    - : float = 4.0

    # second ['a'];;
    Exception: Failure "Not enough values"
```

## 2 The result is a list

### Exercise 2.1 (Arithmetic List)

Write the function arith_list $n$ $a_1$ $r$ which builds the list of first $n$ elements of the arithmetic progression from $a_1$ and of common ratio $r$.

*Examples of applications:*

```
    # arith_list 12 2 1;;
    - : int list = [2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13]
    # arith_list 11 0 3;;
    - : int list = [0; 3; 6; 9; 12; 15; 18; 21; 24; 27; 30]
```

### Exercise 2.2 (Concatenation)

Write a function that concatenates two lists (same as the operator @).

*Example of result:*

```
               # append [1;2;3;4] [5;6];;
               - : int list = [1; 2; 3; 4; 5; 6]
```

## 3 List and Order

### Exercise 3.1 (Growing?)

Write a function that tests whether a list is sorted by increasing order.

### Exercise 3.2 (Deletion)

Write a function that removes the first appearance of an element $x$ (if it is present) from a sorted (in increasing order) list $l$.

### Exercise 3.3 (Insertion at the rank $i$ – *C1 - 11/2020*)

Write the function `insert_nth` *x i list* that inserts the value $x$ at the rank $i$ in the list *list*.
The function has to raise an exception `Invalid_argument` if $i$ is negative or an exception `Failure` if the list is too short.
*Application examples:*

```
# insert_nth 0 5 [1; 2; 3; 4; 5; 6; 7; 8; 9];;
{- : int list = [1; 2; 3; 4; 0; 5; 6; 7; 8; 9]

# insert_nth 0 10 [1; 2; 3; 4; 5; 6; 7; 8; 9];;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 0]

# insert_nth 0 12 [1; 2; 3; 4; 5; 6; 7; 8; 9];;
Exception: Failure "out of bound".

# insert_nth 0 (-2) [1; 2; 3; 4; 5; 6; 7; 8; 9];;
Exception: Invalid_arg "negative rank".
```

### Exercise 3.4 (Penultimate – *C1# 05/2021*)

Write the function `insert_penultimate` *x list* that inserts element $x$ in the penultimate position of list *list*. The function has to raise an exception `Invalid_argument` if the list given as a parameter is empty.

```
# insert_penultimate (-1) [1;8;30;4];;
- : int list = [1; 8; 30; -1; 4]
# insert_penultimate "td" ["caml"];;
- : string list = ["td"; "caml"]
# insert_penultimate (42) [];;
Exception: Invalid_argument "insert_penultimate: empty list".
```

### Exercise 3.5 (Reverse)

Write a function that reverses a list:

1. using the operator `@`;
2. without using the operator `@`.

What do you think of the complexity order of these two functions?

## 4 Two Lists

### Exercise 4.1 (Equals)

Write a function that tests whether two lists are equal.

### Exercise 4.2 (Merge)

Write a function that merges two sorted lists into one.
How to eliminate common elements to both lists?

*Example of result:*

```
# merge [1; 5; 6; 8; 9; 15] [2; 3; 4; 5; 7; 9; 10; 11];;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 15]
```

### Exercise 4.3 (Bonus : Sublist)

Write a function that takes into arguments two lists `l` and `sl` and that returns the number of times the list `sl` is in the list `l`.

```
    # sublist [1; 2] [1; 2] ;;
    - : int = 1
    # sublist [1; 2] [1; 1; 2; 3; 3; 1; 2; 3] ;;
    - : int = 2
    # sublist [1; 2] [2; 1] ;;
    - : int = 0
```

## 5 Small Problems

### Exercise 5.1 (LCM: Using prime factorizations)

Least common multiple of two integers u and v can be computed by

- determining the prime factorizations of u and v
- merge the decompositions of u and v while removing duplicates
- and multiplying the common factors.

Write the function `lcm` (Least Common Multiple) that calculates the lcm of two integers by computing the product of their common factors.
*Examples of result:*

```
        # lcm 8 6;;
        - : int = 24
        # lcm 5 12;;
        - : int = 60
```

### Exercise 5.2 (Bonus: Sequence)

Let the following sequence be:

| | | |
|---|---|---|
| *line 0* : | 1 | is 1 "1" |
| *line 1* : | 11 | is 2 "1" |
| *line 2* : | 21 | is 1 "2" followed by 1 "1" |
| *line 3* : | 1211 | is 1 "1" followed by 1 "2" followed by 2 "1" |
| *line 4* : | 111221 | ... |
| *line 5* : | 312211 | |
| *line 6* : | 13112221 | |

...

Write a function that returns the $n^{th}$ line of this sequence, as an integer list.
*Application examples:*

```
        # sequence 5 ;;
        - : int list = [3; 1; 2; 2; 1; 1]
        # sequence 0 ;;
        - : int list = [1]
```