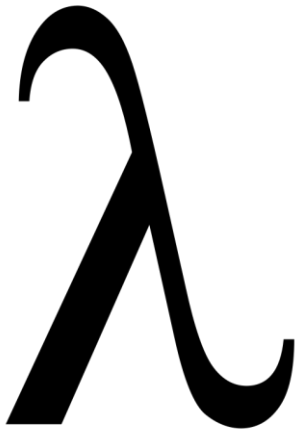

ML Lab 5

Programmazione Funzionale

2024/2025

Università di Trento

Chiara Di Francescomarino

Exercise 4.10

- Suppose that sets are represented by lists. Write a function `powerSet` that takes a list `L`, and produces as output the power set of the list
- If S is a set, the power set of S is the set of all subsets S' such that $S' \subseteq S$
- You can use support functions, if needed
- For instance
  - `powerSet([1,2,3])=[[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]]`
  - `powerSet [#"a",#"c"]=[[], [#"c"], [#"a"], [#"a", #"c"]]`
  - `powerSet nil = [[]]` (do not care about type warning)

Solution 4.10

```
> fun powerSet(nil) = [nil]
    | powerSet(x::xs) =
       powerSet(xs)@insertAll(x,powerSet(xs));
val powerSet = fn: 'a list -> 'a list list


> powerSet [1,2,3];
val it = [[], [3], [2], [2, 3], [1], [1, 3], [1,
2], [1, 2, 3]]:
int list list
```

Exercise 5.1

- Write a short function `thousandthPower` that, given a real `x`, uses `let val` to compute x^{1000}
- For instance,
  - `thousandthPower 1.0 = 1.0`
  - `thousandthPower 1.1 = 2.469932918E41`

Solution 5.1

```
> fun thousandthPower(x:real) =
    let
        val x = x*x*x*x*x*x*x;
        val x = x*x*x*x*x*x*x;
        val x = x*x*x*x*x*x*x
    in
        x*x*x*x*x*x*x*x*x*x
    end;
val thousandthPower = fn: real -> real

> thousandthPower 1.1;
val it = 2.469932918E41: real
```

Exercise 5.2

- Write the `split` function without using a pattern (the tuple) in the `let val` declaration but referencing the components of the tuple
- That is, instead of using the tuple, use the operator for accessing the first and the second item of the recursive call of the split function.

```
> fun split(nil) = (nil,nil)
    | split([a]) = ([a],nil)
    | split (a::b::cs) =
        let
            val(M,N) = split (cs);
        in
            (a::M,b::N)
        end;
val split = fn: 'a list -> 'a list * 'a list
```

Solution 5.2

```
fun split(nil) = (nil,nil)
    | split([a]) = ([a],nil)
    | split (a::b::cs) =
    let
        val x = split (cs);
        val M = #1 x;
        val N = #2 x
    in
        (a::M,b::N)
    end;
val split = fn: 'a list -> 'a list * 'a list

> split [1,2,3,4];
val it = ([1, 3], [2, 4]): int list * int list
```

Exercise 5.3

- Improve the `powerSet` function by using a `let val` declaration and computing the powerset of the tail only once.

```
fun powerSet(nil) = [nil]
   | powerSet(x::xs) =
      powerSet(xs)@insertAll(x,powerSet(xs));
powerSet = fn: 'a list -> 'a list list
```

Solution 5.3

```
> fun powerSet(nil) = [nil]
    | powerSet(x::xs) =
    let
        val L = powerSet(xs)
    in
        L @ insertAll(x,L)
    end;
val powerSet = fn: 'a list -> 'a list list

> powerSet [1,2,3];
val it = [[], [3], [2], [2, 3], [1], [1, 3], [1, 2], [1, 2,
3]]:
int list list
```

Exercise 5.4

- Write a function `sumPairs` that takes a list of pairs of integers, and returns a pair of the sum of each component using the `let val` declaration.
- For instance
  - `sumPairs [(1,2),(3,4),(5,6)] = (9, 12)`
    `sumPairs [] = (0,0)`

Solution 5.4

```
> fun sumPairs (nil) = (0,0)
    | sumPairs ((x,y)::zs) =
    let
        val(z1,z2) = sumPairs(zs)
    in
        (x+z1,y+z2)
    end;

val sumPairs = fn: (int * int) list -> int * int
> sumPairs [(1,2),(3,4),(5,6)];
val it = (9, 12): int * int
```

When you have time

Join this Wooclap event

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
DQDQRX

Exercise 5.5

- Write another version of the function `maxList` (reported below) to find the largest of a list of reals using a `let val` declaration .
- Suggestion: you can think about the maximum of the tail

```
fun maxList([x:real]) = x
   | maxList(x::y::zs) =
      if x<y then maxList(y::zs)
      else maxList(x::zs);
poly: : warning: Matches are not exhaustive.
val maxList = fn: real list -> real
```

Solution 5.5

```
> fun maxList(nil) = 0.0
    | maxList ((x)::xs) =
    let
        val maxTail = maxList (xs)
    in
        if x<maxTail then maxTail else x
    end;
val maxList = fn: real list -> real

> maxList [1.0,4.5,3.2];
val it = 4.5: real
```

Exercise 5.6

- Write an efficient program `doubleExp` that, given as input a real x and nonnegative i, computes x^{2^i}, making only one recursive call.
- For instance,
  - `doubleExp(1.1,3) = 2.14358881`
- Please remember that

$$x^{(2^i)} = x^{2*2^{i-1}} = x^{2^{i-1}+2^{i-1}} = x^{2^{i-1}} * x^{2^{i-1}}$$

Solution 5.6

```
> fun doubleExp(x:real,0) = x
    | doubleExp(x,i) =
        let
            val y = doubleExp(x,i-1)
        in
            y*y
        end;
val doubleExp = fn: real * int -> real

> doubleExp(1.1,3);
val it = 2.14358881: real
```

Exercise 5.7

- Write a function `sumList` that takes a list of integers and returns a pair of the sum of the even positions and the sum of the odd positions using the `let val` declaration.
- For instance
  - `sumList`([1,2,3,4]) = (4,6)
  - `sumList([]) = (0,0)`

Solution 5.7

```
> fun sumList (nil) = (0,0)
    | sumList ([x]) = (0,x)
    | sumList (x::y::zs) =
    let
        val (sumOdd, sumEven) = sumList (zs)
    in
        (x+sumOdd, y+sumEven)
    end;

val sumList = fn: int list -> int * int
> sumList [1,2,3,4,5];
val it = (9, 6): int * int
```

Exercise 5.8

- Write a function `printList` that given as input a list of integers, prints it.
- For instance:
  - `printList nil;`
  - `printList [1,2,3];`

```
1
2
3
```

Solution 5.8

```
> fun printList(nil) = ()
    | printList(x::xs) = (
        print(Int.toString(x));
        print("\n");
        printList(xs)
    );
val printList = fn: int list -> unit

> printList [1,2,3];
1
2
3
val it = (): unit
```

Exercise 5.9

- Write a function `comb` to compute $\binom{n}{m}$, while printing n, m and the result.
- You can use the factorial (write an auxiliary function `factorial`) to compute the function $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
- For instance
  - `comb 5 2;`

```
n is 5
m is 2
Result is 10
```

Solution 5.9

```
fun factorial 0 = 1
| factorial n = n * factorial(n-1);
fun comb n m =
    (
    print ("n is ");
    print(Int.toString(n));
    print ("\n");
    print ("m is ");
    print(Int.toString(m));
    print ("\n");
    print ("Result is ");
    print (Int.toString (factorial(n) div (factorial(m) * factorial(n-
    m))));
    print ("\n")
    );
comb 5 2;
n is 5
m is 2
Result is 10
val it = (): unit
```

Exercise 5.10

- Write a function `printXs` that given an integer n, print 2^n `X`s

- For instance
  - `printXs 3;`

    `XXXXXXXX`

Solution 5.10

```
> fun makelist 0 = "X"
    | makelist n = makelist (n-1) ^ makelist (n-1);
val makelist = fn: int -> string
> fun printXs n = print(makelist n);
val printXs = fn: int -> unit

> printXs 3;
XXXXXXXXval it = (): unit
```

- Or, alternatively

```
> fun print_2n (0) = print("X")
      |print_2n (n) = (
              print_2n (n-1);
              print_2n (n-1)
              );
```

Exercise 5.11

- Write expressions to
  1. Open a file “zap” for reading
  2. Read 5 characters from the instream in1
  3. Read a line of text from the instream in1
  4. Find the first character waiting on the in1, without consuming it
  5. Read the entire file from instream in1
  6. Close the file whose instream is in1

Solution 5.11

- Write expressions to
  1. Open a file “zap” for reading
     ```
     val in1 = TextIO.openIn(“zap”);
     ```
  2. Read 5 characters from the instream in1
     ```
     TextIO.inputN(in1,5);
     ```
  3. Read a line of text from the instream in1
     ```
     TextIO.inputLine(in1)
     ```
  4. Find the first character waiting on the in1, without consuming it
     ```
     TextIO.lookaehead(in1);
     ```
  5. Read the entire file from instream in1
     ```
     TextIO.input(in1)
     ```
  6. Close the file whose instream is in1
     ```
     TextIO.closeIn (in1)
     ```

Exercise 5.12

- Write a function `getList(filename)` that reads a file, extracts the words (without space characters), and transforms the file in a list of words (without space characters).
- For instance, if the file `helloworld` contains

  Hello beautiful world!

  Bye!

  ```
  getList(helloworld) = ["Hello", "beautiful", "world",
  "Bye!"]
  ```

- Hint: first write a function `getWord(in)` that extracts a word (without spaces) from a `TextIO.instream in` and then put them in a list. You can use support functions

Solution 5.12

```
(* test if a character is white space *)
> fun white(" ") = true
    | white("\t") = true
    | white("\n") = true
    | white("\r") = true
    | white(_) = false;

(* read one word *)
> fun getWord(file) =
    if TextIO.endOfStream(file) then ""
    else
        let
                val c = TextIO.inputN(file,1)
        in
                if white(c) then ""
                else c^getWord(file)
    end;
```

Solution 5.12 (continue)

```
(*test if a string is empty*)
> fun is_empty("")=true
        |is_empty(_)=false;
> fun getList1(file) = (* read all words from an instream *)
    if TextIO.endOfStream(file) then nil
        else
                let
                        val w = getWord(file);
                        val tail = getList1(file)
                in
                        if is_empty(w) then tail
                        else w::tail
                end;
(* read all words from a file given the file name *)
> fun getList(filename) = getList1(TextIO.openIn(filename));
```