λ

ML Lab 2

Programmazione Funzionale

2024/2025

Università di Trento

Chiara Di Francescomarino

Tutoring

- Starting from next Wednesday (March 19th)
- Every Wednesday 15:30 – 16:30 in pc A201
- On Wednesday April 2nd 15:30 – 16:30 in pc A202
- No tutoring time on April 9th because of the ICT days

- For those of you attending the English course who also wants to attend the tutoring time, please fill in the form in Moodle or drop me an email every time you need it (Friday 11:30 – 12:30)

Exercise L2.1

- Write a function `third` that computes the third element of a list (it doesn't have to work properly on shorter lists).

- For instance,

```
third [2,3,4] = 4
third [2,3,4,5] = 4
```

Solution L2.1

```
> fun third (l) = hd (tl(tl(l)));
val third = fn: 'a list -> ‘a

> third [2,3,4];
val it = 4: int
> third [2,3,4,5];
val it = 4: int
> third [2,3];
Exception- Empty raised
```

Exercise L2.2

- Write a function `reverse` that reverses a tuple of length 3
- For instance,

```
reverse (1,2,3) = (3,2,1)
reverse (1.0,2,"a") = ("a",2,1.0)
```

Solution L2.2

```
> fun reverse(a,b,c) = (c,b,a);
val reverse = fn: 'a * 'b * 'c -> 'c * 'b * 'a

> reverse (1,2,3);
val it = (3, 2, 1): int * int * int
> reverse (1.0,2,"a");
val it = ("a", 2, 1.0): string * int * real
```

Exercise L2.3

- Write a function `thirdchar` that finds the string contatining the third character of a string (it doesn't have to work properly on shorter strings)
- For instance

```
thirdchar "abcd" = "c"
```

Solution L2.3

```
> fun thirdchar(s) = str(third(explode s));
val thirdchar = fn: string -> string

> fun thirdchar(s) = str(hd(tl(tl(explode
s))));
val thirdchar = fn: string -> string


> thirdchar "abcd";
val it = "c": string
```

Exercise L2.4

- Write a function `cycle` that cycles a list once, i.e., convert `[a1,...,an]` to `[a2,...,an,a1]`. It doesn't have to work on the empty list.
- For instance
  - `cycle [1,2,3,4] = [2,3,4,1]`
  - `cycle [1] = [1]`

Solution L2.4

```
> fun cycle (l) = tl(l) @ [hd(l)];
val cycle = fn: 'a list -> 'a list

> cycle [1,2,3,4];
val it = [2, 3, 4, 1]: int list
> cycle [1,2];
val it = [2, 1]: int list
> cycle [1];
val it = [1]: int list
```

Exercise L2.5

- Write a function `min_max_pair` that, given 3 integers (a tuple), produce a pair consisting of the smallest and the largest value among the 3 integers
- For instance
  - `min_max_pair (1,2,3) = (1,3)`
  - `min_max_pair(3,4,2) = (2,4)`

Solution L2.5

```
> fun min3 (a,b,c) = if a<b then if a<c then a else c
                     else
                     if b<c then b else c;
val min3 = fn: int * int * int -> int
> fun max3 (a,b,c) = if a>b then if a>c then a else c
                     else
                     if b>c then b else c;
val max3 = fn: int * int * int -> int
> fun min_max_pair (a,b,c) = (min3(a,b,c),max3(a,b,c));
val min_max_pair = fn: int * int * int -> int * int

> min_max_pair (1,2,3);
val it = (1, 3): int * int
> min_max_pair(3,4,2);
val it = (2, 4): int * int
```

Exercise L2.6

- Write a function `sort` that, given three integers (a tuple), produce a list of them in sorted order
- For instance
  - `sort (1,2,3) = [1,2,3]`
  - `sort (1,3,2) = [1,2,3]`

Solution L2.6

```
> fun medium (a,b,c) = if a<b then if b<c then b else if a<c then c
else a else if b>c then b else if a<c then a else c;
val medium = fn: int * int * int -> int
> fun sort (a,b,c) = min(a,b,c)::medium(a,b,c)::[max(a,b,c)];
val sort = fn: int * int * int -> int list

> fun sort (a,b,c) = [min3(a,b,c)] @ [if a<b then if b<c then b else if
a<c then c else a else if c<b then b else if a<c then a
else c] @ [max3(a,b,c)];
val sort = fn: int * int * int -> int list

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

Exercise L2.7

- Write a function `rnd` that rounds a real number to the nearest decimal (10th) digit
- For instance
  - `rnd(2.56) = 2.6`
  - `rnd (5.678) = 5.7`
  - `rnd (3.3) = 3.3`
  - `rnd (4.128) = 4.1`

Solution L2.7

```
> fun rnd (r:real) = real (round(r *10.0)) / 10.0;
val rnd = fn: real -> real

> rnd (5.678);
val it = 5.7: real
> rnd 5.628;
val it = 5.6: real
```

Exercise L2.8

- Write a function `rem` that, given a list, removes the second element. It doesn't need to work on lists shorter than 2.
- For instance
  - `rem [1,2,3,4] = [1,3,4]`
  - `rem [1,2] = [1]`

Solution L2.8

```
>fun rem l = hd(l) :: tl(tl(l));
val rem = fn: 'a list -> 'a list

> rem [1,2,3,4];
val it = [1, 3, 4]: int list
> rem [1,2];
val it = [1]: int list
```