



- Introduction and overview
- Basic types, definitions and functions
- Basic data structures
- More advanced data structures
- Higher order functions
- Exceptions, input/output and imperative constructs

► Modules and data abstraction

Table of Contents

Guests

Structuring software with modules

Week 6 Échéance le déc 12, 2016 at 23:30 UTC

Information hiding

Week 6 Échéance le déc 12, 2016 at 23:30 UTC

Case study: A module for dictionaries

Week 6 Échéance le déc 12, 2016 at 23:30 UTC

Functors

Week 6 Échéance le déc 12, 2016 at 23:30 UTC

Modules as compilation units

- Project

REMOVE ELEMENTS FROM DICTIONARIES (20/20 points)

The following code is the program explained during the video sequence except that we have modified the interface `DictSig` a little bit. Now, it is possible to `remove` a key from a dictionary.

1. Update the code to have it accepted by the type-checker.

THE GIVEN PRELUDE

```
module type DictSig = sig
  type ('key, 'value) t
  val empty : ('key, 'value) t
  val add : ('key, 'value) t -> 'key -> 'value -> ('key, 'value) t
  exception NotFound
  val lookup : ('key, 'value) t -> 'key -> 'value
  val remove : ('key, 'value) t -> 'key -> ('key, 'value) t
end ;;
```

YOUR OCAML ENVIRONMENT

```
1 module Dict : DictSig = struct
2   type ('key, 'value) t =
3     | Empty
4     | Node of ('key, 'value) t * 'key * 'value * ('key, 'value) t
5
6   let empty = Empty
7
8   let rec add d k v =
9     match d with
10    | Empty -> Node (Empty, k, v, Empty)
11    | Node (l, k', v', r) ->
12      if k = k' then Node (l, k, v, r)
13      else if k < k' then Node (add l k v, k', v', r)
14      else Node (l, k', v', add r k v)
15
16   exception NotFound
17
18   let rec lookup d k =
19     match d with
20     | Empty ->
21       raise NotFound
22     | Node (l, k', v', r) ->
23       if k = k' then v'
24       else if k < k' then lookup l k
25       else lookup r k
26
27   let rec find_max = function
28     | Empty -> assert false
29     | Node (_, k, v, Empty) -> (k, v)
30     | Node (_, k, v, r) -> find_max r;;
31
32   let rec remove d k = match d with
33     | Empty ->
```

[Evaluate >](#)[Switch >>](#)[Typecheck](#)[Reset Templ](#)[Full-screen |](#)[Check & Sa](#)

Exercise complete (click for details)

20 pts

Completed, 20 pts

► Exercise 1: Dict

Found Dict with compatible type.

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.remove d "Eggplant" in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.remove d "Zucchini" in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.add d "Zucchini" 3 in
let d = Dict.remove d "Zucchini" in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.add d "Zucchini" 3 in
let d = Dict.add d "Banana" 1 in
let d = Dict.remove d "Eggplant" in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.add d "Zucchini" 3 in
let d = Dict.add d "Banana" 1 in
let d = Dict.remove d "Apple" in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Radish" 2 in
let d = Dict.add d "Orange" 4 in
let d = Dict.add d "Zucchini" 5 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Banana" 4 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Apple" 7 in
let d = Dict.add d "Salad" 2 in
let d = Dict.remove d "Salad" in
let d = Dict.add d "Tomato" 0 in
let d = Dict.add d "Bean" 9 in
let d = Dict.remove d "Apple" in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Carrot" 2 in
let d = Dict.add d "Eggplant" 2 in
let d = Dict.remove d "Carrot" in
let d = Dict.remove d "Eggplant" in
let d = Dict.add d "Orange" 8 in
let d = Dict.add d "Bean" 3 in
let d = Dict.remove d "Orange" in
let d = Dict.add d "Eggplant" 4 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Salad" 8 in
let d = Dict.remove d "Salad" in
let d = Dict.add d "Tomato" 6 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Carrot" 10 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Apple" 4 in
let d = Dict.add d "Banana" 7 in
let d = Dict.remove d "Banana" in
let d = Dict.add d "Zucchini" 7 in
d
Correct dictionary returned.
Computing the following sequence:
let d = Dict.empty in
let d = Dict.add d "Radish" 0 in
let d = Dict.add d "Tomato" 10 in
let d = Dict.add d "Banana" 2 in
let d = Dict.remove d "Radish" in
let d = Dict.remove d "Tomato" in
let d = Dict.remove d "Banana" in
let d = Dict.add d "Salad" 7 in
d
```

1 pt

1 pt

1 pt

1 pt

1 pt

1 pt

1 pt

1 pt

1 pt

1 pt

```
let d = Dict.add d "Carrot" 1 in
let d = Dict.remove d "Carrot" in
let d = Dict.add d "Apple" 1 in
let d = Dict.add d "Eggplant" 5 in
let d = Dict.remove d "Eggplant" in
let d = Dict.add d "Bean" 2 in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Zucchini" 4 in
let d = Dict.add d "Orange" 6 in
let d = Dict.add d "Carrot" 3 in
let d = Dict.remove d "Orange" in
let d = Dict.add d "Bean" 9 in
let d = Dict.add d "Apple" 6 in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Orange" 10 in
let d = Dict.add d "Banana" 0 in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Eggplant" 2 in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Salad" 1 in
let d = Dict.remove d "Salad" in
let d = Dict.add d "Zucchini" 9 in
let d = Dict.add d "Tomato" 3 in
let d = Dict.add d "Radish" 5 in
let d = Dict.remove d "Radish" in
let d = Dict.add d "Eggplant" 0 in
let d = Dict.add d "Salad" 6 in
d
```

Correct dictionary returned.

1 pt

Computing the following sequence:

```
let d = Dict.empty in
let d = Dict.add d "Tomato" 8 in
let d = Dict.remove d "Tomato" in
let d = Dict.add d "Banana" 1 in
let d = Dict.add d "Bean" 10 in
let d = Dict.remove d "Banana" in
let d = Dict.add d "Apple" 6 in
let d = Dict.add d "Zucchini" 5 in
let d = Dict.remove d "Zucchini" in
d
```

Correct dictionary returned.

1 pt

[A propos](#)

[Aide](#)

[Contact](#)

[Conditions générales d'utilisation](#)

[Charte utilisateurs](#)

[Politique de confidentialité](#)

[Mentions légales](#)