

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

Ocaml Datatypes

<https://powcoder.com>

Add WeChat powcoder
CSI 3120

Amy Felty

University of Ottawa

OCaml So Far

- We have seen a number of basic types:
 - int
 - float
 - char
 - string
 - bool
- We have seen a few structured types:
 - pairs
 - tuples
 - options
 - lists
- In this lecture, we will see some more general ways to define our own new types and data structures

<https://powcoder.com> Type Abbreviations

- We have already seen some type abbreviations:

Add WeChat powcoder

```
type point = float * float
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

<https://powcoder.com> Type Abbreviations

- We have already seen some type abbreviations:

Add WeChat powcoder

```
type point = float * float
```

- These abbreviations can be helpful documentation:

<https://powcoder.com>

```
let distance (p1:point) (p2:point) : float =
  let square x = x *. x in
  let (x1, y1) = p1 in
  let (x2, y2) = p2 in
  sqrt (square (x2 -. x1) +. square (y2 -. y1))
```

- But they add nothing of *substance* to the language
 - they are **equal** in every way to an existing type

<https://powcoder.com> Type Abbreviations

- We have already seen some type abbreviations:

Add WeChat powcoder

```
type point = float * float
```

- As far as OCaml is concerned, you could have written:

Add WeChat powcoder

```
let distance (p1:float*float)
              (p2:float*float) : float =
  let square x = x *. x in
  let (x1,y1) = p1 in
  let (x2,y2) = p2 in
  sqrt (square (x2 -. x1) +. square (y2 -. y1))
```

- Since the types are equal, you can *substitute* the definition for the name wherever you want
 - we have not added any new data structures

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

DATA TYPES

https://powcoder.com

Data types

- OCaml provides a general mechanism called a **data type** for defining new data structures that consist of many alternatives

Add WeChat powcoder

```
type my_bool = Tru | Fal
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

a value with type **my_bool**
is one of two things:

- Tru, or
- Fal

read the " | " as "or"

https://powcoder.com

Data types

- OCaml provides a general mechanism called a **data type** for defining new data structures that consist of many alternatives

Add WeChat powcoder

```
type my_bool = Tru | Fal
```

Assignment Project Exam Help

https://powcoder.com

Tru and Fal are called
"constructors"

Add WeChat powcoder

a value with type **my_bool**
is one of two things:

- Tru, or
- Fal

read the " | " as "or"

https://powcoder.com

Data types

- OCaml provides a general mechanism called a **data type** for defining new data structures that consist of many alternatives

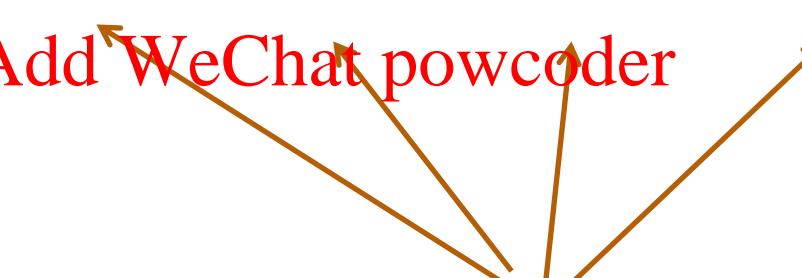
Add WeChat powcoder

```
type my_bool = True | False
```

Add WeChat powcoder

```
type color = Blue | Yellow | Green | Red
```

Add WeChat powcoder



there's no need to stop
at 2 cases; define as many
alternatives as you want

https://powcoder.com

Data types

- OCaml provides a general mechanism called a **data type** for defining new data structures that consist of many alternatives

Add WeChat powcoder

```
type my_bool = True | False
```

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

- Creating values:

```
let b1 : my_bool = True  
let b2 : my_bool = False  
let c1 : color = Yellow  
let c2 : color = Red
```

use constructors to create values

https://powcoder.com

Data types

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

Add WeChat powcoder

```
let c1 : color = Yellow
```

```
let c2 : color = Red
```

Assignment Project Exam Help

- Using data type values:

```
let print_color (c:color) : unit =
  match c with
  | Blue ->
  | Yellow ->
  | Green ->
  | Red ->
```

use pattern matching to
determine which color
you have; act accordingly

<https://powcoder.com>

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

[Add WeChat powcoder](#)

```
let c1 : color = Yellow
```

```
let c2 : color = Red
```

Assignment Project Exam Help

- Using data type values:

```
let print_color (c:color) : unit =
  match c with
  | Blue -> print_string "blue"
  | Yellow -> print_string "yellow"
  | Green -> print_string "green"
  | Red -> print_string "red"
```

https://powcoder.com Data types

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

Add WeChat powcoder

```
let c1 : color = Yellow
```

```
let c2 : color = Red
```

Assignment Project Exam Help

- Using data type values:

```
let print_color (c:color) : unit =
  match c with
  | Blue -> print_string "blue"
  | Yellow -> print_string "yellow"
  | Green -> print_string "green"
  | Red -> print_string "red"
```

Why not just use strings to represent colors instead of defining a new type?

https://powcoder.com Data types

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

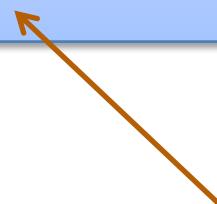
Add WeChat powcoder

oops!:

```
let print_color (c:color) : unit =
  match c with
    | Blue -> print_string "blue"
    | Yellow -> print_string "yellow"
    | Red -> print_string "red"
```

Add WeChat powcoder

Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
Green



https://powcoder.com Data types

Assignment Project Exam Help

```
type color = Blue | Yellow | Green | Red
```

Add WeChat powcoder

oops!:

```
let print_color (c:color) : unit =
  match c with
  | Blue -> print_string "blue"
  | Yellow -> print_string "yellow"
  | Red -> print_string "red"
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
Green

OCaml's datatype mechanism allow you to create types
that contain *precisely* the values you want!

Data Types Can Carry Additional Values

- Data types are more than just enumerations of constants:

Add WeChat powcoder

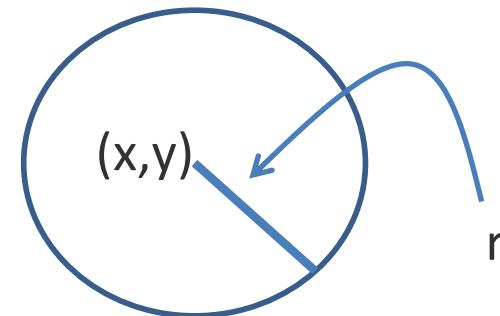
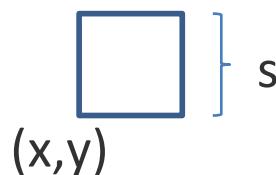
```
type point = float * float
```

```
type simple_shape =
    Circle of point * float
```

```
| Square of point * float
```

Add WeChat powcoder

- Read as: a **simple_shape** is either:
 - a **Circle**, which contains a **pair** of a **point** and **float**, or
 - a **Square**, which contains a **pair** of a **point** and **float**



Data Types Can Carry Additional Values

- Data types are more than just enumerations of constants:

Add WeChat powcoder

```
type point = float * float
```

```
type simple_shape =
  Circle of point * float
  | Square of point * float
```

```
let origin : point = (0.0, 0.0)
```

```
let circ1 : simple_shape = Circle (origin, 1.0)
let circ2 : simple_shape = Circle ((1.0, 1.0), 5.0)
let square : simple_shape = Square (origin, 2.3)
```

Data Types Can Carry Additional Values

- Data types are more than just enumerations of constants:

Add WeChat powcoder

```
type point = float * float
```

```
type simple_shape =  
  Circle of point * float
```

```
| Square of point * float
```

Add WeChat powcoder
<https://powcoder.com>

```
let simple_area (s:simple_shape) : float =  
  match s with  
  | Circle (_, radius) -> 3.14 *. radius *. radius  
  | Square (_, side) -> side *. side
```

<https://powcoder.com>

- Data types are more than just enumerations of constants:

Add WeChat powcoder

```
type point = float * float
```

```
type simple_shape =
  Circle of point * float
  | Square of point * float
```

```
let simple_area (s:simple_shape) : float =
  match s with
  | Circle (_, radius) -> 3.14 *. radius *. radius
  | Square (_, side) -> side *. side
```

```
type my_shape = point * float
```

```
let simple_area (s:my_shape) : float =
  (3.14 *. radius *. radius) ?? or ?? (side *. side)
```

[More General Shapes](https://powcoder.com)

Assignment Project Exam Help

```
type point = float * float
```

```
type shape = Add WeChat powcoder
```

```
    Square of float
```

```
| Ellipse of float * float
```

```
| RtTriangle of float * float
```

```
| Polygon of point list
```

Assignment Project Exam Help

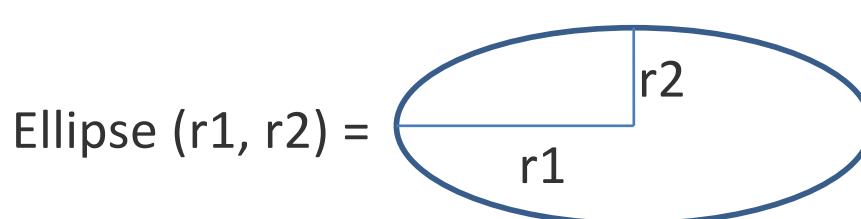
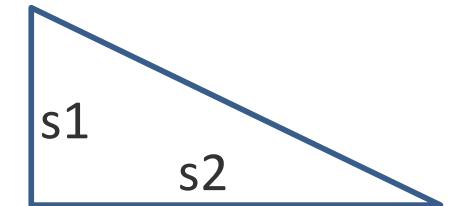
<https://powcoder.com>

Add WeChat powcoder

Square s =

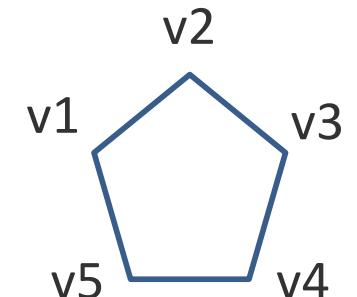


RtTriangle (s1, s2) =



Ellipse (r1, r2) =

Polygon [v1; ...; v5] =



More General Shapes

Assignment Project Exam Help

```
type point = float * float
```

```
type radius = float
```

```
type side = float
```

```
type shape =
```

Square of side

| Ellipse of radius * radius

| RtTriangle of side * side

| Polygon of point list

Assignment Project Exam Help

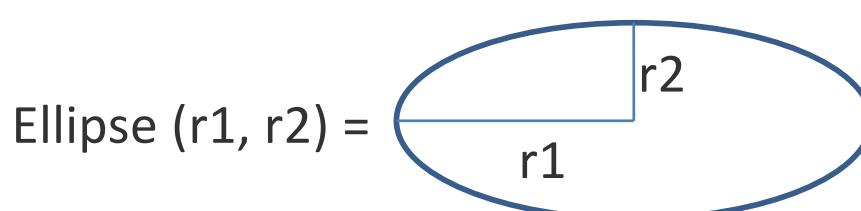
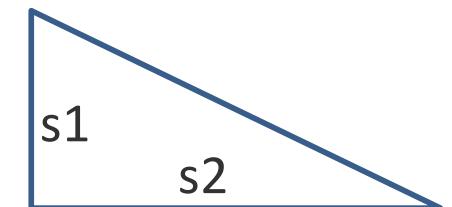
<https://powcoder.com>

Type abbreviations can aid readability

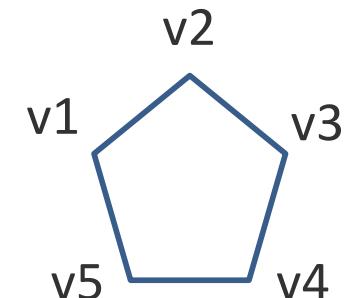
Square s =



RtTriangle (s1, s2) =



RtTriangle [v1; ...; v5] =



More General Shapes

Assignment Project Exam Help

```
type point = float * float
type radius = float
type side = float
```

Square builds a shape
from a single side

```
type shape =
  | Square of side
  | Ellipse of radius * radius
  | RtTriangle of side * side
  | Polygon of point list
```

Assignment Project Exam Help

Ellipse builds a shape
from a pair of radii

```
let sq   : shape = Square 17.0
let ell  : shape = Ellipse (1.0, 2.0)
let rt   : shape = RtTriangle (1.0, 1.0)
let poly : shape = Polygon [(0., 0.); (1., 0.); (0.; 1.)]
```

RtTriangle builds a shape
from a pair of sides

they are all shapes;
they are constructed in
different ways

Polygon builds a shape
from a list of points
(where each point is itself a pair)

Assignment Project Exam Help

Add WeChat powcoder

https://powcoder.com

More General Shapes

Assignment Project Exam Help

```
type point = float * float
```

```
type radius = float
```

```
type side = float
```

```
type shape =
```

```
    Square of side
```

```
| Ellipse of radius * radius
```

```
| RtTriangle of side * side
```

```
| Polygon of point list
```

Assignment Project Exam Help

a data type also defines
a pattern for matching

Add WeChat powcoder

```
let area (s : shape) : float
```

```
match s with
```

```
| Square s ->
```

```
| Ellipse (r1, r2) ->
```

```
| RtTriangle (s1, s2) ->
```

```
| Polygon ps ->
```

<https://powcoder.com>

More General Shapes

Assignment Project Exam Help

```
type point = float * float
```

```
type radius = float
```

```
type side = float
```

```
type shape =
```

- | Square of side

- | Ellipse of radius * radius

- | RtTriangle of side * side

- | Polygon of point list

Assignment Project Exam Help

<https://powcoder.com>

a data type also defines
a pattern for matching

```
let area (s : shape) : float
```

```
match s with
```

- | Square s ->

- | Ellipse (r1, r2) ->

- | RtTriangle (s1, s2) ->

- | Polygon ps ->



Square carries a value
with type **float** so **s** is
a pattern for float values

RtTriangle carries a value
with type **float * float**
so **(s1, s2)** is a pattern
for that type

More General Shapes

Assignment Project Exam Help

```
type point = float * float
```

```
type radius = float
```

```
type side = float
```

```
type shape =
```

- | Square of side

- | Ellipse of radius * radius

- | RtTriangle of side * side

- | Polygon of point list

Assignment Project Exam Help

a data type also defines
a pattern for matching

Add WeChat powcoder

```
let area (s : shape) : float
```

```
match s with
```

- | Square s -> s *. s

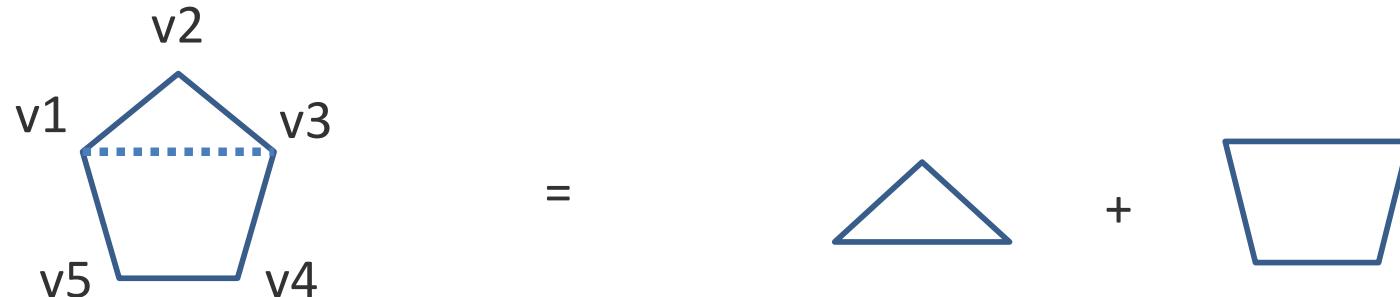
- | Ellipse (r1, r2) -> pi *. r1 *. r2

- | RtTriangle (s1, s2) -> s1 *. s2 /. 2.

- | Polygon ps -> ???

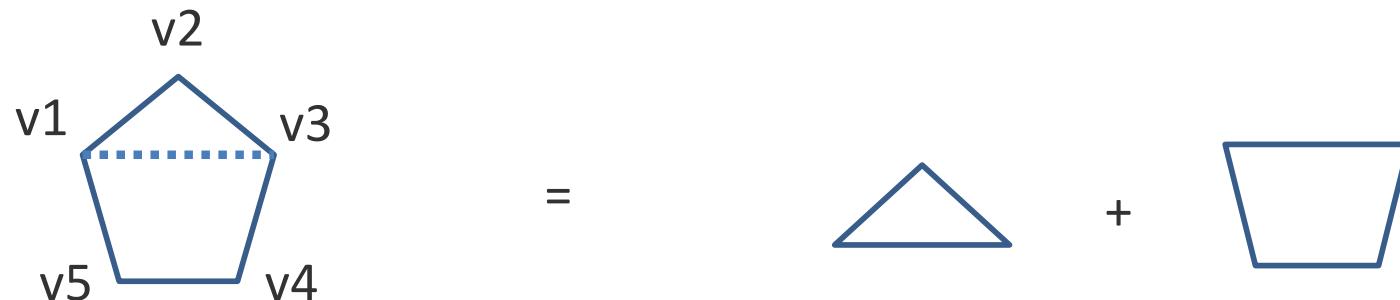
Computing Area

- How do we compute polygon area?
- For convex polygons:
 - Case: the polygon has fewer than 3 points:
 - it has 0 area! (it is a line or a point or nothing at all)
 - Case: the polygon has 3 or more points:
 - Compute the area of the triangle formed by the first 3 vertices
 - Delete the second vertex to form a new polygon
 - Sum the area of the triangle and the new polygon



<https://powcoder.com> Computing Area

- How do we compute polygon area?
- For convex polygons:
 - Case: the polygon has fewer than 3 points:
 - it has 0 area! (it is a line or a point or nothing at all)
 - Case: the polygon has 3 points:
 - Compute the area of the triangle formed by the first 3 vertices
 - Delete the second vertex to form a new polygon
 - Sum the area of the triangle and the new polygon
- Note: This is a beautiful inductive algorithm:
 - the area of a polygon with n points is computed in terms of a smaller polygon with only $n-1$ points!



https://powcoder.com Computing Area

Assignment Project Exam Help

```
let area (s : shape) : float =
  match s with
  | Square s -> s *. s
  | Ellipse (r1, r2) -> r1 *. r2
  | RtTriangle (s1, s2) -> s1 *. s2 /. 2.
  | Polygon ps -> poly_area ps
```

Add WeChat powcoder

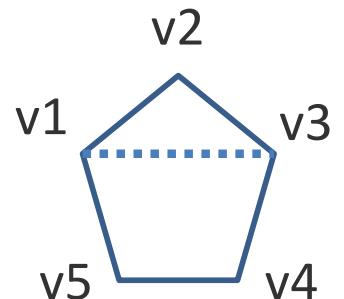
Assignment Project Exam Help

This pattern says the list has at least 3 items

```
let rec poly_area (ps : point list) : float =
  match ps with
  | p1 :: p2 :: p3 :: tail ->
    tri_area p1 p2 p3 +. poly_area (p1 :: p3 :: tail)
  | _ -> 0.
```

https://powcoder.com

Add WeChat powcoder



=



+



<https://powcoder.com> Computing Area

Assignment Project Exam Help

```
let tri_area (p1:point) (p2:point) (p3:point) : float =
  let a = distance p1 p2 in
  let b = distance p2 p3 in
  let c = distance p3 p1 in
  let s = 0.5 *. (a +. b +. c) in
  sqrt (s *. (s -. a) *. (s -. b) *. (s -. c))
```

Assignment Project Exam Help

```
let rec poly_area (ps: point list): float =
  match ps with
  | p1 :: p2 :: p3 :: tail > Add WeChat powcoder
    tri_area p1 p2 p3 +. poly_area (p1 :: p3 :: tail)
  | _ -> 0.
```

```
let area (s : shape) : float =
  match s with
  | Square s -> s *. s
  | Ellipse (r1, r2) -> pi *. r1 *. r2
  | RtTriangle (s1, s2) -> s1 *. s2 /. 2.
  | Polygon ps -> poly_area ps
```

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

INDUCTIVE DATA TYPES

Inductive data types

- We can use data types to define inductive data
- A binary tree is:
 - a Leaf containing no data
 - a Node containing a key, a value, a left subtree and a right subtree

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Inductive data types

- We can use data types to define inductive data
- A binary tree is:
– a Leaf containing no data
– a Node containing a key, a value, a left subtree and a right subtree

Assignment Project Exam Help

```
type key = int
type value = string
type tree =
    Leaf
  | Node of key * value * tree * tree
```

Inductive data types

Assignment Project Exam Help

```
type key = int  
type value = string
```

```
type tree =  
  Leaf  
 | Node of key * value * tree * tree
```

<https://powcoder.com>

```
let rec insert (t:tree) (k:key) (v:value) : tree =
```

Add WeChat powcoder

Inductive data types

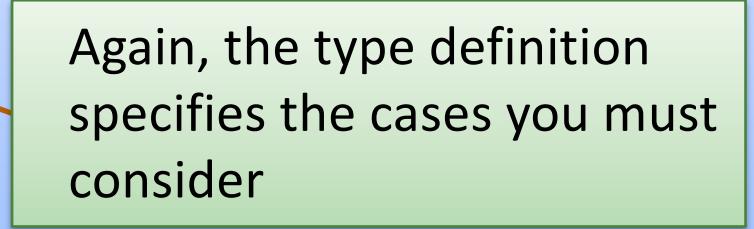
Assignment Project Exam Help

```
type key = int  
type value = string
```

```
type tree =  
  Leaf  
 | Node of key * value * tree * tree
```

<https://powcoder.com>

```
let rec insert (t:tree) (k:key) (v:value) : tree =  
  match t with  
    | Leaf ->  
    | Node (k', v', left, right) ->
```



Again, the type definition specifies the cases you must consider

Inductive data types

Assignment Project Exam Help

```
type key = int  
type value = string
```

```
type tree =  
  Leaf  
 | Node of key * value * tree * tree
```

[Assignment Project Exam Help](https://powcoder.com)

```
let rec insert (t:tree) (k:key) (v:value) : tree =  
  match t with  
    | Leaf -> Node (k, v, Leaf, Leaf)  
    | Node (k', v', left, right) ->
```

Inductive data types

Assignment Project Exam Help

```
type key = int
type value = string
```

```
type tree =
  Leaf
  | Node of key * value * tree * tree
```

[Assignment Project Exam Help](https://powcoder.com)

```
let rec insert (t:tree) (k:key) (v:value) : tree =
  match t with
  | Leaf -> Node (k, v, Leaf, Leaf)
  | Node (k', v', left, right) ->
    if k < k' then
      Node (k', v', insert left k v, right)
    else if k > k' then
      Node (k', v', left, insert right k v)
    else
      Node (k, v, left, right)
```

Inductive data types

Assignment Project Exam Help

```
type key = int
type value = string
```

```
type tree =
  Leaf
  | Node of key * value * tree * tree
```

[Assignment Project Exam Help](https://powcoder.com)

```
let rec insert (t:tree) (k:key) (v:value) : tree =
  match t with
    | Leaf -> Node (k, v, Leaf, Leaf)
    | Node (k', v', left, right) ->
        if k < k' then
          Node (k', v', insert left k v, right)
        else if k > k' then
          Node (k', v', left, insert right k v)
        else
          Node (k, v, left, right)
```

Inductive data types

Assignment Project Exam Help

```
type key = int
type value = string
```

```
type tree =
  Leaf
  | Node of key * value * tree * tree
```

[Assignment Project Exam Help](https://powcoder.com)

```
let rec insert (t:tree) (k:key) (v:value) : tree =
  match t with
    | Leaf -> Node (k, v, Leaf, Leaf)
    | Node (k', v', left, right) ->
        if k < k' then
          Node (k', v', insert left k v, right)
        else if k > k' then
          Node (k', v', left, insert right k v)
        else
          Node (k, v, left, right)
```

Inductive data types: Another Example

- We can use the type "Int" to represent natural numbers
 - but that is kind of broken: it also contains negative numbers
 - we have to use a dynamic test and a default value:

```
let double(n : Int) : Int =  
  if n < 0 then  
    0  
  else  
    double_hat n
```

Inductive data types: Another Example

- We can use the type "int" to represent natural numbers
 - but that is kind of broken: it also contains negative numbers
 - we have to use a dynamic test and a default value:
 - or raise an exception

```
let double (n : int) : int =  
  if n < 0 then  
    raise Failure ("negative input!")  
  else  
    double_hat n
```

- it would be nice if there was a way to define the natural numbers **exactly**, and use OCaml's type system to guarantee no client ever attempts to double a negative number

<https://powcoder.com> Inductive data types

- Recall, a natural number n is either:
 - zero, or [Add WeChat powcoder](#)
 - $m + 1$
- We use a data type to represent this definition exactly:

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

[Add WeChat powcoder](#)

Inductive data types

- Recall, a natural number n is either:
 - zero, or [Add WeChat powcoder](https://powcoder.com)
 - $m + 1$
- We use a data type to represent this definition exactly:

Assignment Project Exam Help

```
type nat = Zero | Succ of nat
```

<https://powcoder.com>

[Add WeChat powcoder](#)

Inductive data types

- Recall, a natural number n is either:
 - zero, or [Add WeChat powcoder](https://powcoder.com)
 - $m + 1$
- We use a data type to represent this definition exactly:

Assignment Project Exam Help

```
type nat = Zero | Succ of nat  
https://powcoder.com  
let rec nat_to_int (n : nat) : int =  
  match n with Add WeChat powcoder  
    Zero -> 0  
  | Succ n -> 1 + nat_to_int n
```

Inductive data types

- Recall, a natural number n is either:
 - zero, or [Add WeChat powcoder](https://powcoder.com)
 - $m + 1$
- We use a data type to represent this definition exactly:

Assignment Project Exam Help

```
type nat = Zero | Succ of nat  
https://powcoder.com  
let rec nat_to_int (n : nat) : int =  
  match n with Add WeChat powcoder  
    | Zero -> 0  
    | Succ n -> 1 + nat_to_int n  
  
let rec double_nat (n : nat) : nat =  
  match n with  
    | Zero -> Zero  
    | Succ m -> Succ (Succ (double_nat m))
```

<https://powcoder.com>

- OCaml data types: a powerful mechanism for defining complex data structures:
[Assignment Project Exam Help](#)
[Add WeChat powcoder](#)
 - They are precise
 - contain exactly the elements you want, not more elements
 - They are general
[Assignment Project Exam Help](#)
 - recursive, non-recursive (mutually recursive and polymorphic)
 - The type checker helps you detect errors
 - missing cases in your functions

<https://powcoder.com>

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

PARAMETERIZED TYPE DEFINITIONS

<https://powcoder.com>

```
type ('key, 'val) tree =
  Leaf
  | Node of 'key * 'val * ('key, 'val) tree * ('key, 'val) tree
```

Add WeChat powcoder

```
type 'a inttree = (int, 'a) tree
```

type istree = string inttree

<https://powcoder.com>

Note: istree is an abbreviation for (int, string) tree

Add WeChat powcoder

General form:

definition:

type 'x f = body

use:

arg f

A more conventional notation
would have been (but is not ML):

definition:

type f x = body

use:

f arg

Take-home Message

- Think of parameterized types like functions:
 - a function that takes a type as an argument
 - produces a type as a result
- Theoretical basis:
 - System F-omega
 - a typed lambda calculus with general type-level functions as well as value-level functions