# Programming Languages and Techniques (CIS120)

Lecture 9

February 7, 2014

Academic Integrity Discussion

Finite Maps

## Announcements

- Homework 3 is due *TUESDAY* at 11:59:59pm

- Read Chapter 9 of lecture notes
- Read collaboration policy on syllabus

- Midterm 1
  - Scheduled in class on *Friday, February 21st*
  - Contact me if you need to take the make-up exam
  - More details to follow!

# Homework 2

- Feedback:
  - This was much more difficult than the first assignment. (x5)
  - Not too hard.
  - It was a little tough at times, but regardless good fun to work on. Also taught me a lot about working with lists and trees in OCaml.
  - I liked how this last problem really made you go back through the outputs of all of the prior functions.
  - So... tired... want... sleep...
- Note: we don't connect feedback directly to submissions. Put notes to your TAs in your comments!
- Timespent:
  - avg=8.5, max=20, min=1, n=122

## Academic Integrity

- From lecture 1
- Submitted homework must be *your individual work*

clear

Not OK:

- Copying someone else's code

OK / encouraged:

- "High level" discussions of concepts

Penn's code of academic integrity:
http://www.vpul.upenn.edu/osl/acadint.html

Problematic

1. unclear
2. stronger than necessary

## What I Really Care About

- The homework assignments in CIS120 are 80% of the value of the course (though only 50% of the grade)
- To get this value, people need to do the assignments themselves
- **Principle**: Every bit of every assignment that you turn in should come from *your brain*

Is the following OK according to the "from your own brain" principle?

> Fred found parts of an OCaml implementation of binary search trees online somewhere and handed it in as his own solution.

1. OK
2. marginal
3. not OK

Is the following OK according to the "from your own brain" principle?

> Fred found parts of an OCaml implementation of binary search trees in the course slides/lecture notes and handed it in as his own solution.

1. OK
2. marginal
3. not OK

Is the following OK according to the "from your own brain" principle?

> Fred had three midterms this week and didn't have time to look at the homework. Jane had finished early, and she emailed Fred her working version, and Fred handed this in.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Fred’s homework was mostly working, but he couldn’t get add_ancestor_labels to behave correctly. He copied just this method from Jane’s working solution.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Frank and Frieda worked on the homework together, with Frieda typing while Frank looked over her shoulder and gave suggestions. At the end, they had one solution, which they both turned in.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Frank and Frieda worked on the homework together, with Frieda typing while Frank looked over her shoulder and gave suggestions.  At the end, Frank went away and wrote out a new solution from scratch; since he’d already been through the answers once, this didn’t take much time.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Mark and Mary worked on the homework side by side, each working on their own laptop, stopping to help each other when one person got stuck.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Joe had been stuck on a nasty bug for hours.  Mary saw how frustrated he was and sat down to help him.  Together, they found and fixed the problem.

1. OK
2. marginal
3. not OK

Is the following OK according to the “from your own brain” principle?

> Joe had been stuck on a nasty bug for hours. Mary saw how frustrated he was and sat down to help him. She immediately noticed the place where Joe had made a mistake and said “Here, just change this <= to < and it will work.”

1. OK
2. marginal
3. not OK

## Academic Integrity

- Refined rules:

Not OK:
- Copying / sharing of code
- Looking at someone else's code while typing your own
- Giving answers (as opposed to teaching someone how to find them)
- Other people's code on your computer
- …

OK, with care:
- debugging assistance
- clarification of concepts related to homework

OK / encouraged:
- "High level" discussions of concepts from lecture

## Abstract types

## Abstract Types

- Example from Wednesday: sets!
- Different programming languages have different ways of letting you define abstract types
- At a minimum, this means providing:
  - A way to specify (write down) an interface
  - A means of hiding implementation details (*encapsulation*)
- In OCaml:
  - Interfaces are specified using a *signature* or *interface*
  - Encapsulation is achieved because the interface can *omit* information
    - type definitions
    - names and types of auxiliary functions
  - Clients *cannot* mention values not named in the interface

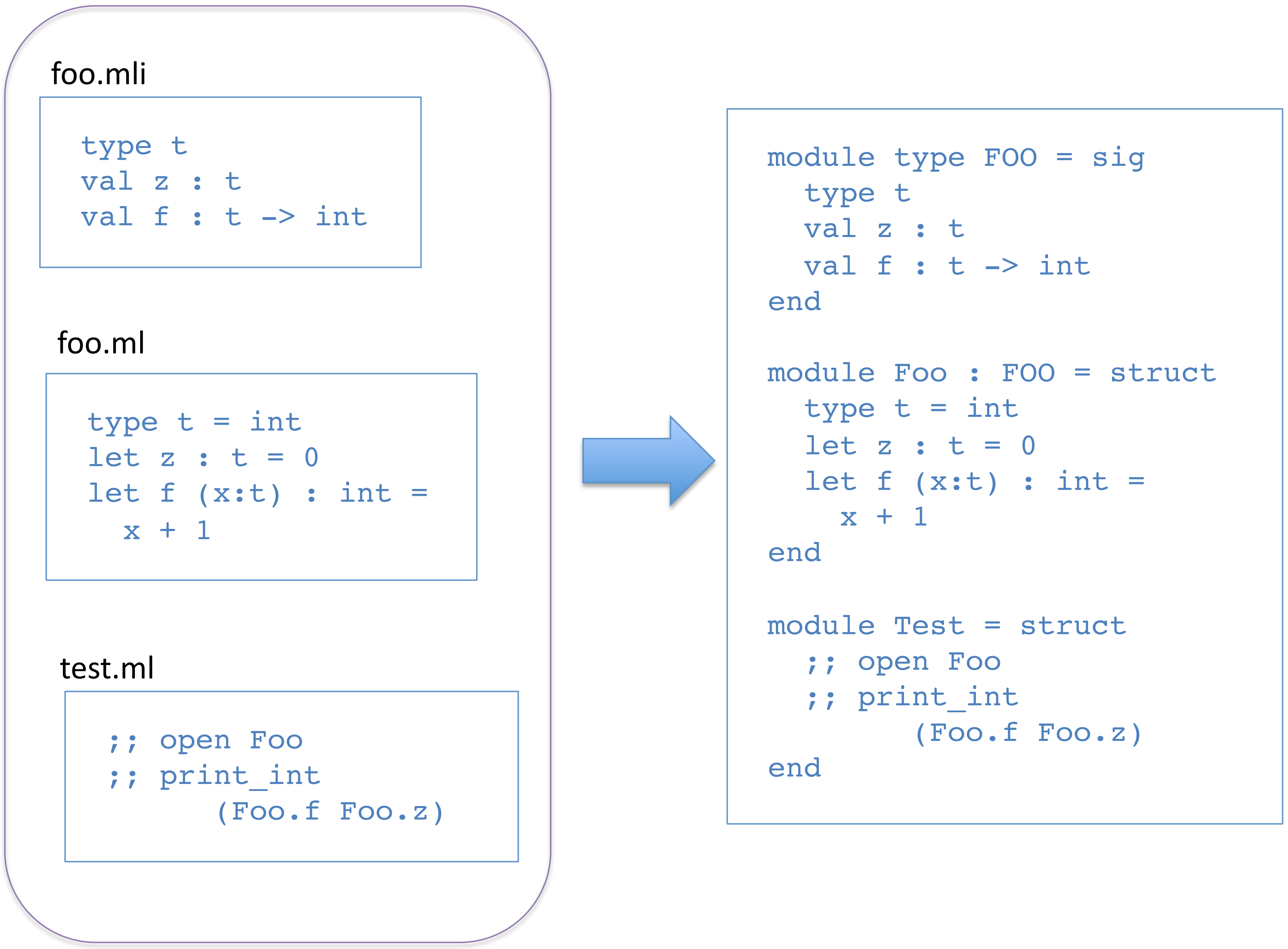
# Modules and signatures

```
module type Set = sig

   type 'a set
   val empty  : 'a set
   …

end

module MySet : Set = struct
  type 'a tree =
   | Empty
   | Node of 'a tree * 'a * 'a tree
  type 'a set = 'a tree

  let empty : 'a set = Empty
  …
end
```

## .ml and .mli files

- You’ve already been using signatures and modules in OCaml.

- A series of type and `val` declarations stored in a file `foo.mli` is considered as defining a signature `FOO`
- A series of top-level definitions stored in a file `foo.ml` is considered as defining a module `Foo`



foo.mli

```
type t
val z : t
val f : t -> int
```

foo.ml

```
type t = int
let z : t = 0
let f (x:t) : int =
  x + 1
```

test.ml

```
;; open Foo
;; print_int
     (Foo.f Foo.z)
```

```
module type FOO = sig
  type t
  val z : t
  val f : t -> int
end

module Foo : FOO = struct
  type t = int
  let z : t = 0
  let f (x:t) : int =
    x + 1
end

module Test = struct
  ;; open Foo
  ;; print_int
        (Foo.f Foo.z)
end
```

## Finite Map Demo

Using module signatures to preserve
data structure invariants

mymap.ml

mymapTest.ml

## Motivating Scenario

- Suppose you were writing some course-management software and needed to look up the lab section for a student given the student's PennKey?
  - Students might add/drop the course
  - Students might switch lab sections
  - Students should be in only *one* lab section

- How would you do it?

# Finite Maps

- A *finite map* (a.k.a. *dictionary*)*,* is a collection of *bindings* from distinct *keys* to *values*.
  - Operations to add & remove bindings, test for key membership, look up a value by its key
- Example: a `(string, int) map` might map a PennKey to the lab section.
  - The map type is generic in two arguments
- Like sets, finite maps appear in many settings:
  - map domain names to IP addresses
  - map words to their definitions (a dictionary)
  - map user names to passwords
  - map game character unique identifiers to dialog trees
  - …