


<p>Outline Intro Tools Standard ML Exercises</p> <div style="background-color: #800000; color: white; padding: 10px; margin: 20px auto; width: 80%;"> <h2 style="margin: 0;">CSE341 – Section 1</h2> <p style="margin: 0;">Emacs, SML Mode, Shadowing, Error Messages</p> </div> <p style="margin: 20px auto;">Cody A. Schroeder</p> <p style="margin: 20px auto;">January 10th, 2013</p>	<p>Outline Intro Tools Standard ML Exercises</p> <ol style="list-style-type: none"> 1 Intro 2 Tools <ul style="list-style-type: none"> • Emacs • The REPL 3 Standard ML <ul style="list-style-type: none"> • Shadowing Variables • Reading Errors • Boolean Operators 4 Exercises <ul style="list-style-type: none"> • Exercises • Solutions
Cody A. Schroeder CSE341 – Section 1	Cody A. Schroeder CSE341 – Section 1

<p>Outline Intro Tools Standard ML Exercises</p> <h2 style="margin: 0;">Intro</h2> <div style="background-color: #800000; color: white; padding: 5px; margin: 10px auto; width: 80%;"> <p>Hello! I'm Cody!!!</p> </div> <ul style="list-style-type: none"> • One of the two main section leaders (with Eric). <ul style="list-style-type: none"> • We'll probably alternate weeks. • Remember: Cody is the fancy one. → • I'm one of the 5th year masters students. • Previously TA'd many times including 341 twice. • This is one of my favorite classes! 	<p>Outline Intro Tools Standard ML Exercises</p> <h2 style="margin: 0;">Emacs</h2> <ul style="list-style-type: none"> • Our EDITOR of choice this quarter! • Not necessarily required but recommended. • Plenty of cheat sheets and tutorials: <ul style="list-style-type: none"> • Stanford Cheat Sheet, Ref Card, OLD UW Tutorial, etc. • Also the staff is <i>almost</i> always available for help! <div style="background-color: #800000; color: white; padding: 5px; margin: 10px auto; width: 80%;"> <h3 style="margin: 0;">Demo Time</h3> <ul style="list-style-type: none"> • The basics! • Any questions? </div>
Cody A. Schroeder CSE341 – Section 1	Cody A. Schroeder CSE341 – Section 1

Outline Intro Tools Standard ML Exercises **Errors**
Outline Intro Tools Standard ML Exercises Reading Errors Boolean Operators

The REPL

- Read-Eval-Print-Loop
- Meant for iterative development and real-time testing.
- Usually load a file using **use**, test functions, edit source, restart REPL, and repeat.
- Very powerful tool for convenience.

Note: It's dangerous to call **use** more than once in the same REPL.

Shadowing of Variable Bindings

SML Example

```
1 val x = "Hello World!";
2 val x = 2; (* Is this allowed? *)
3 val res = x*2; (* Is this 4 or a type error? *)
```

- There is **no assignment** in SML.
- However, a variable name can be bound multiple times.
- When looking up a variable, the latest binding in the current scope is used.
- Any previous bindings are said to be **shadowed**.
 - A similar shadowing effect occurs in other languages like **Java**.
- This is the reason calling **use** more than once on the same file can cause problems.

Cody A. Schroeder CSE341 – Section 1
Cody A. Schroeder CSE341 – Section 1

Outline Intro Tools Standard ML Exercises Reading Errors Boolean Operators
Outline Intro Tools Standard ML Exercises Shadowing Variables Boolean Operators

Shadowing of Variable Bindings (cont.)

Another shadowing example

```
1 fun absOriginal x = abs x; (* Save abs function . *)
2 fun abs (x,y) = (absOriginal x, absOriginal y);
```

FAQ

- Is it ever possible to use a shadowed variable? **Yes! And no...**
- It can be possible to uncover a shadowed variable when the latest binding goes out of scope.

Using a Shadowed Variable

```
1 val x = "Hello World!";
2 fun add1 (x : int) = x+1; (* Shadow x in function body *)
3 val y = add1 2;
4 val z = x^"!1!"; (* "Hello World!!!" *)
```

Dealing with error messages from SML

We've Got Errors...

```
1 val x = 34;
2 y = x + 1;
3 val z = if y > 4 then 34 else x < 4;
4 val q = if y > 0 then 0;
5 val a = -5;
6 val w = 0;
7 val fun = 34;
8 val v = x / w;
```

```
1 val x = 34;
2 val y = x + 1; (* Missing val *)
3 val z = if y > 4 then false else x < 4; (* if/else typing error *)
4 val q = if y > 0 then 0 else 1; (* Missing else branch *)
5 val a = -5; (* Used binary - *)
6 val w = 0;
7 val funn = 34; (* fun is a keyword *)
8 val v = x div (w+1); (* Can't (/) ints *)
```

Cody A. Schroeder CSE341 – Section 1
Cody A. Schroeder CSE341 – Section 1

<p>Outline Intro Tools Standard ML Exercises Shadowing Variables Reading Errors</p> <h2>Boolean Operators</h2> <p>The Operators</p> <ul style="list-style-type: none"> • andalso (same as Java's <code>&&</code>) • orelse (same as Java's <code> </code>) • not (just a function) <ul style="list-style-type: none"> • Why can not be a function while the others cannot? <ul style="list-style-type: none"> • Because andalso and orelse may not evaluate both its left and right sides. They short-circuit evaluation. • Be careful to always use andalso instead of and. • and is completely different. We will get back to it later. <p>Cody A. Schroeder CSE341 – Section 1</p>	<p>Outline Intro Tools Standard ML Exercises Solutions</p> <h2>Exercises</h2> <p>Write the xor function.</p> <p>Given three ints, return their min and max in a pair.</p> <p>Write a function that computes the n^{th} Fibonacci number.</p> <p>Implement a function that, given an real x, results in a real equal to the equation $x^2 - x/2 + 5$. Calculate $f(-2)$ with it.</p> <p>Cody A. Schroeder CSE341 – Section 1</p>
--	---

Outline Intro Tools Standard ML Exercises Exercises **Solutions**

Solutions

```

1 fun xor1 (b1 : bool, b2 : bool) = if b1 then not b2 else b2;
2 fun xor2 (b1 : bool, b2 : bool) = (b1 orelse b2) andalso
3   not (b1 andalso b2);

1 fun minmax (a : int, b : int, c : int) =
2   (Int.min (a, Int.min (b,c)), Int.max (a, Int.max (b,c)));
3 (* Or write a bunch of nested ifs. *)

1 fun fib (n : int) = if n < 2 then n else fib (n-1) + fib (n-2);

1 fun f (x : real) = x*x - x/2.0 + 5.0;

```

Cody A. Schroeder CSE341 – Section 1