Outline

Mutual Recursion

Modules in ML

Currying
Mutual Recursion

Even or odd?

It may be desirable to define a function \( f \) that calls a function \( g \), but also allow \( g \) to call \( f \):

\[
\text{fun is_even x = if x = 0 then true else is_odd (x - 1)}
\]

\[
\text{fun is_odd x = if x = 0 then false else is_even (x - 1)}
\]

What could go wrong here?

At the time we're defining \( \text{is_even} \), \( \text{is_odd} \) is undefined.
Mutual Recursion

Even or odd?

It may be desirable to define a function $f$ that calls a function $g$, but also allow $g$ to call $f$:

```plaintext
fun is_even x = 
  if x = 0 
  then true 
  else is_odd (x - 1)

fun is_odd x = 
  if x = 0 
  then false 
  else is_even (x - 1)
```
Mutual Recursion

Even or odd?

It may be desirable to define a function \( f \) that calls a function \( g \), but also allow \( g \) to call \( f \):

```sh
fun is_even x = 
  if x = 0
  then true
  else is_odd (x - 1)

fun is_odd x = 
  if x = 0
  then false
  else is_even (x - 1)
```

What could go wrong here?
Mutual Recursion
Even or odd?

It may be desirable to define a function $f$ that calls a function $g$, but also allow $g$ to call $f$:

```plaintext
fun is_even x = 
  if x = 0 
  then true 
  else is_odd (x - 1)

fun is_odd x = 
  if x = 0 
  then false 
  else is_even (x - 1)
```

What could go wrong here?
At the time we’re defining `is_even`, `is_odd` is undefined
Allow is_even to be higher order, so that we can pass is_odd to it:

```plaintext
fun is_even f x = 
  if x = 0
  then true
  else f (x - 1)

fun is_odd x = 
  if x = 0
  then false
  else is_even is_odd (x - 1)
```
Mutual Recursion

Even or odd?

Allow `is_even` to be higher order, so that we can pass `is_odd` to it:

```
fun is_even f x =
    if x = 0
    then true
    else f (x - 1)

fun is_odd x =
    if x = 0
    then false
    else is_even is_odd (x - 1)
```

Can we do better?
Mutual Recursion

Even or odd?

ML allows for mutual recursion with the `and` keyword

```ml
fun is_even x = 
  if x = 0
  then true
  else is_odd (x - 1)

and is_odd x = 
  if x = 0
  then false
  else is_even (x - 1)
```
Mutual Recursion

Even or odd?

ML allows for mutual recursion with the `and` keyword

```ml
fun is_even x = 
  if x = 0
  then true
  else is_odd (x - 1)

and is_odd x = 
  if x = 0
  then false
  else is_even (x - 1)
```

With `and`, we can also define a mutually recursive `datatype` too
Modules in ML
Abstraction

We saw modules in lecture:

```plaintext
signature MATHLIB =
  sig
  val fact : int -> int
  val half_pi : real
  val doubler : int -> int
  end

structure MyMathLib :> MATHLIB =
  struct
  fun fact x = ...
  val half_pi = Math.pi / 2.0
  fun doubler x = x * 2
  end
```

1. Good for organization and managing namespaces
2. Helpful for maintaining invariants
3. Especially **helpful** for hiding implementation details
Invariants
Some Examples

1. Order of operations (e.g. insert query before searching)
2. Data kept in good shape (e.g. Rational from lecture only allows reduced fractions)
3. Following policy (e.g. don’t allow shipping requests without a purchase order)
Currying
Lots of Examples

** Code will be available on the course website