CSE 341 Fall 2011 Section 10: Final Exam Review

- **Disclaimer:**
  This is a selection of a few topics we spent little or no time on in homework.
  It does not necessarily reflect what will be on the exam.

Macros!

- Write a MUPL macro to double a number without defining any functions. The argument should be evaluated only once.
- Write a Racket macro following the same guidelines.
- How can programmers using the Racket or MUPL macros distinguish them from functions?
- Can the Racket and MUPL macros behave differently? Is this true of all such "translation-equivalent" pairs of Racket/MUPL macros?

Suppose we decide to add multiple inheritance to Ruby. What is one issue we need to address with respect to this code?

```
class Vehicle
  def drive
    ...
    steer(dir)
    ...
    move
    ...
  end
end

class WheeledVehicle < Vehicle
  ...
  def steer(dir)
    @frontwheels.each {|w| w.turn(dir)}
  end
  def move
    @wheels.each {|w| w.rotate}
  end
end

class RudderedVehicle < Vehicle
  ...
  def steer(dir)
    @rudder.move(dir)
  end
end

# BoatCar inherits from both WheeledVehicle
# and RudderedVehicle.
class BoatCar < WheeledVehicle, RudderedVehicle
   ...
end
```

Add function and record subtyping to ML. Describe the standard subtyping rules.

- Function subtyping: _________ in the argument and _________ in the result
- Record subtyping: use only width subtyping of records.
  - Width subtyping means:

Note, some of the parentheses are not needed, but to separate issues of currying, they are made explicit.

Add function and record subtyping to ML.
Does each ans_ typecheck? Why?

- type A = { b : bool }
- type B = { b : bool, j : int }
- val x : A
- val y : B
- val f : A -> B
- val g : B -> B
- val h : (A -> A) -> B
- val i : (B -> A) -> (A -> B)

- val ans1 = f x
- val ans2 = f y
- val ans3 = f (f x)
- val ans4 = f (g x)
- val ans5 = f (g y)
- val ans6 = h f
- val ans7 = h g
- val ans8 = (i h) x
- val ans9 = (i g) y
- val ans10 = (i f) x

Note, some of the parentheses are not needed, but to separate issues of currying, they are made explicit.

Add function and record subtyping to ML.
Does each ans_ typecheck? Why?

- type A = { b : bool }
- type B = { b : bool, j : int }

B <: A

- val x : A
- val y : B
- val f : A -> B
- val g : B -> B
- val h : (A -> A) -> B
- val i : (B -> A) -> (A -> B)

val ans1 = f x	**yes**
val ans2 = f y	**yes**
val ans3 = f (f x)	**yes**
val ans4 = f (g x)	**no**
val ans5 = f (g y)	**yes**
val ans6 = h f	**yes**
val ans7 = h g	**no**
val ans8 = (i h) x	**no**
val ans9 = (i g) y	**yes**
val ans10 = (i f) x	**yes**

Note, some of the parentheses are not needed, but to separate issues of currying, they are made explicit.