CIS 120 Quiz 4

February 11–12, 2015

Name: ____________________________________________________________

PennKey (e.g. mkizner): __________________________ Section: _____________

Indicate the section you’re registered for, even if you’re attending a different section.
1. Consider the following module definition.

```ocaml
module M = struct
  type grade = int
  let good_style (g: grade) (pts: int) : grade =
    let new_grade = g + pts in
    min 10 new_grade
  let bad_style : grade = 0
  let to_int (g: grade) : int = g
  let from_int (pts: int) : grade = pts
end
```

As the naming suggests, the designer of the module wants to use the type `grade` to represent a numeric grade for an assignment; grades should range from 0 to 10, inclusive. Evaluate whether each of the following signatures for `M` could be used to maintain this invariant.

(a) type grade = int
   val bad_style : grade
   val to_int : grade -> int
   Does this signature...?
   □ Prevent clients from breaking the invariant
   □ Allow clients to break the invariant
   □ Fail to type-check

(b) type grade
    val bad_style : grade
    val to_int : grade -> int
    Does this signature...?
    □ Prevent clients from breaking the invariant
    □ Allow clients to break the invariant
    □ Fail to type-check

(c) type grade
    val good_style : grade -> int -> grade
    val bad_style : grade
    val to_int : grade -> int
    val from_int : int -> grade
    Does this signature...?
    □ Prevent clients from breaking the invariant
    □ Allow clients to break the invariant
    □ Fail to type-check

(d) type grade
    val good_style : grade -> int -> grade
    val ok_style : grade -> int -> grade
    val bad_style : grade
    val to_int : grade -> int
    val from_int : int -> grade
    Does this signature...?
    □ Prevent clients from breaking the invariant
    □ Allow clients to break the invariant
    □ Fail to type-check

(e) type grade
    val good_style : grade -> int -> grade
    val bad_style : grade
    val to_int : grade -> int
    Does this signature...?
    □ Prevent clients from breaking the invariant
    □ Allow clients to break the invariant
    □ Fail to type-check