1. C and C++ use pointer arithmetic, but Java, being a descendant of C-like languages (descendant of ALGOL languages by extension), doesn't. Why is that?

2. Are Java enumerations considered a primitive type? What are some advantages to using enumerations?

3. What does the ordinal() method do in Java Enums, and how could it be useful?

4. Write down the types of the following in OCaml:

```
let tripleFloat x = 3.0*.x;;
let thrice f x = f(f(f(x)));;
let composition f g x = f(g(x));;
let div x y = x/y;;
let triple3 = thrice tripleFloat;;
```

5. In OCaml, you cannot use an integer and a floating point argument directly in the same expression, as you may be able to in Java, or Python, for example. How would you use explicit type-casting to get around turning an integer into a floating-point number, and vice-versa?

6. One way of showing how Lambda calculus is "used" in a language like OCaml is the `let e1 in e2` structure. For example, `let x = 5 in let y = 3 in x+y;;` is just like (\x.\y.(x+y)) (5) (3) in lambda calculus (where "\" stands for the

lambda symbol). Try making similar conversions with the following OCaml code statements:

a. `let x = 4 in let y = 12 in y/x;;`
b. `let x = 3 in let y = 10 in let z = 5 in (x*y)/z;;`
c. `let f x = x + 3 in let y = 5 in f y;;`

7. Is pattern matching in OCaml more like “If-Else” statements, or more like “Case” statements? For example, take this function:

```
let rec sumList l =
     match l with
     | [] -> 0
     | h::t -> h + sumList t
;;
```