CSE 341 - Programming Languages
Midterm - Autumn 2012

Your Name:

(for recording grades):

1. (max 8)

2. (max 5)

Total (max 60):

3. (max 4)

4. (max 6)

5. (max 12)

6. (max 8)

7. (max 6)

8. (max 6)

9. (max 5)

You can bring a maximum of 2 (paper) pages of notes. No laptops, tablets, or smart phones.

1. (8 points) Suppose that we have a `filter` function in Haskell, defined as follows:

```
filter f [] = []
filter f (x:xs) | f x       = x : filter f xs
                | otherwise = filter f xs
```

For example, `filter even [1,2,3,4]` evaluates to `[2,4]`.

Circle each type declaration that is a correct type for `filter`. (Not necessarily the most general type, just a correct one.)

```
filter :: (Int -> Bool) -> [Int] -> [Int]

filter :: a -> [a] -> [a]

filter :: (a -> b) -> [a] -> [b]

filter :: (a -> Bool) -> [a] -> [a]

filter :: (Bool -> Bool) -> [Bool] -> [Bool]

filter :: (Eq a) => (a -> Bool) -> [a] -> [a]
```

Which of the above types, if any, is the most general type for `filter`?

2. (5 points) What are the first 6 elements in the following Haskell infinite list?

```
mystery = 0 : (map (\x -> 10-x) mystery)
```

3. (4 points) What are the free variables in each of the following Racket expressions?

```
(lambda (x) (if (= x 1) (+ y z) 100))
```

```
(let ((x 5) (y x)) (+ x y))
```

```
(lambda (a) (+ b (let ((b 10)) (+ a b))))
```

```
(let ((s '(1 2 3))) (map f s))
```

4. (6 points) Convert the following Haskell action into an equivalent one that doesn't use do.

```
echo = do
    putStr "type in a number between 1 and 10: "
    n <- readLn
    putStrLn ("Double your number is " ++ show (2*n))
```

5. (12 points) Consider the function `rss` that finds the square root of the sum of the squares of a list of numbers.

   (a) Write a Racket definition of `rss`. You can define a helper function if needed, and you can use built-in Racket functions like `sqrt` and `map`. However, Racket doesn't have a built-in function to find the sum of a list of numbers.

   (b) Write a point-free Haskell definition of `rss`. Again, you can use built-in Haskell functions, which conveniently include `sum` as well as `sqrt` and `map`. Also give the most general type for this function.
   Hint: the type of `sqrt` is
   ```
   Floating a => a -> a
   ```
   where `Floating` is a subclass of `Num`. Partial credit for a correct but non-point-free answer.

6. (8 points) Define an `average` function in Racket that finds the average of two numbers.

Now define a curried version of `average` in Racket.

7. (6 points) What is the result of evaluating the following Racket expressions?

```
(let ((x 1)
      (y 2)
      (z 5))
  (let* ((x (+ x 10))
         (y x))
    (+ x y z)))
```

```
(let ((x 1)
      (y 2)
      (z 5))
  (let ((x (+ x 10))
        (y x))
    (+ x y z)))
```

```
(let* ((x 10)
       (f (lambda (y) (* x y)))
       (x 20))
  (f x))
```

8. (6 points) Consider the following MUPL program.

```
(eval-prog
 (mlet "x" (int 100)
       (mlet "plus1"
             (fun #f "x" (add (var "x") (int 1)))
             (call (var "plus1") (add (var "x") (var "x"))))))
```

For the following questions, write out the environments as lists, in exactly the form used by the MUPL interpreter. If you need to refer to the closure for `plus1` in your answers, you can abbreviate it as `<plus1 closure>` and receive full credit, although it's fine to write it out completely if you prefer.

(For partial credit, write down the environment in some other understandable form.)

(a) What is the environment bound in the closure for the `plus1` function?

(b) What is the environment that MUPL uses when evaluating the body of the `plus1` function when it is called in the above program?

(c) What is the environment that MUPL uses when evaluating the actual parameter to the call to the `plus1` function?

9. (5 points) Suppose the following Racket code is evaluated. How many times is `squid` printed? How many times is `octopus` printed?

```
(define d1 (delay (print "squid ") (+ 3 4)))
(define d2 (delay (print "octopus ") (+ 8 8)))
(force d1)
(force d1)
```