CSE 341: Section 5

1. What are the types of the following x1, x2, … x5? Some might have type errors:

```
b = True

x1 = if b then putStrLn "hi" else return ()

x2 = if b then putStrLn "squid"  else return "octopus"

x3 = if b then "squid" else "octopus"

x4 = if b then "squid" else return ()

x5 = do
     putStr "testing"
     x <- readLn
     return (not x)
```

2. Give a recursive definition of a list `doubles` **whose first element is 10, and whose n th element is twice the n− 1 st, i.e., [10, 20, 40, 80, 160, 320, ...]. To do this, write a helper function doubles_from that takes a parameter n and returns a list of all the doubles starting at n.**

3. Give yet another non-recursive definition of `doubles` **using the built-in function** `iterate` **from the Haskell prelude. This is defined as follows:**

```
iterate :: (a -> a) -> a -> [a]
iterate f x = x : iterate f (f x)
```

4. Define a Haskell list `dollars` that is the infinite list of amounts of money you have every year, assuming you start with $100 and get paid 5% interest, compounded yearly. (Ignore inflation, deflation, taxes, bailouts, the possibility of total economic collapse, and other such details.) So dollars should be equal to [100.0, 105.0, 110.25, ...]

5. Desugar the following actions:

```
lion = do
  putStrLn "What is the color of your mane?"
  color <- getLine
  putStrLn $ "Rawr, nice " ++ color ++ " mane"
```

```
parity_repl = do
  putStrLn "Enter a number"
  n <- readLn
  case odd n of
    True -> putStrLn $ (show n) ++ " is odd"
    False -> putStrLn $ (show n) ++ " is even"
  parity_repl
```

```
map_reduce = do
   A. putStrLn "Enter a unary mapping operation"
   B. op <- getLine
   C. putStrLn "Enter a unary reducing operation"
   D. reduce <- getLine
   E. putStrLn "Enter a list to evaluate"
   F. lst <- getLine
   G. let expr = "foldr1 (" ++ reduce ++ ") $ map (" ++ op ++ ") " ++
      lst
           in evaluate expr
```