CIS 425

Fall 2019

Assignment 9(CPS + Haskell)

December 2019

Problem 1: Lazy Lists Revisited

In Assignment 5, we saw how to implement a lazy list in SML. In Haskell, a lazy list requires no special effort, since *all* data structures are lazy by default. In particular, the built-in list type is lazy.

1. Define in Haskell an infinite list called *code* that is simply a never-ending sequence of ones: $1, 1, 1, 1, 1, \ldots$;

   Solution:

   ```
   ones :: [Int]
   ones = 1 : ones
   ```

   □

2. Write a Haskell function *intList* n that will create a sequence of integers from n to infinity: $n, n+1, n+2, \ldots$ (You may **not** use the special built-in list syntax for this; build the list using only the cons operator *(:)*)

   Solution:

   ```
   intList :: Int -> [Int]
   intList n = n : intList (n + 1)
   ```

   □

3. Write a Haskell function *takeN* that returns the first n elements from a list. (Do not use any standard functions for this.) For example,

   ```
   takeN 4 (intList 10)
   ```

   should evaluate to:

   ```
   [10, 11, 12, 13]
   ```

   Solution:

   ```
   takeN :: Int -> [a] -> [a]
   takeN 0 xs     = []
   takeN n (x:xs) = x : takeN (n - 1) xs
   ```

   □

Problem 2: Stream Equations

1. Define in Haskell the list of all even positive integers and the list of all odd positive integers.

```
evens :: [Int]
evens =
```

and

```
odds :: [Int]
odds =
```

Solution:

```
evens :: [Int]
evens = map (\x -> 2*x) (intList 0)

odds :: [Int]
odds = map (\x -> 2*x + 1) (intList 0)
```

□

2. Define a merge function in Haskell that takes two ordered lists and returns the resulting merged list, in order. For instance,

```
merge [1,2,3] [4,5,6]
```

```
[1,4,2,5,3,6]
```

```
merge :: [Int] -> [Int] -> [Int]
```

Does the call

```
merge evens odds
```

terminate? Explain why or why not in a few sentences. What about

```
length (merge evens odds)
```

Solution:

```
merge :: [a] -> [a] -> [a]
merge [] as         = as
merge as []         = as
merge (a:as) (b:bs) = a : b : merge as bs
```

merge evens odds does terminate because even though **evens** and **odds** are infinite we don't need to evaluate them all at once. **length** on an infinite list also does not terminate. □

3. Write each of the sequences below as one or more Haskell streams (infinite lists). You may use the *merge* function defined above.

   (a) $0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, \ldots$
   Solution:

   ```
   map (\x -> x ^ 3) (intList 0)
   ```

   □

   (b) $1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, \ldots$
   Solution:

   ```
   map (\x -> 3^x) (intList 0)
   ```

   □

   (c) $0, 0, 1, 1, 2, 4, 3, 9, 4, 16, 5, 25, 6, 36, 7, 49, \ldots$
   Solution:

   ```
   merge (intList 0) (map (^2) (intList 0))
   ```

   □

   (d) The negative numbers
   Solution:

```
map (0-) (intList 1)
```

□

For example, the sequence consisting of all zeros can be described as:

```
zeroes :: [Int]
zeroes = 0 : zeroes
```

Alternatively, a list can be described using a **list comprehension**:

```
[n + 1 | n <- [1,2,3]]
```

evaluates to

```
[2,3,4]
```