Programming in Haskell

S P Suresh

http://www.cmi.ac.in/~spsuresh

Lecture 3

August 16, 2017

Example: gcd

- Euclid's algorithm for *gcd(a,b)*, assume *a >= b*
  - If *b* is *0*, define the answer to be *a*
  - Otherwise, *gcd(a,b) = gcd(b, mod a b)*

```
gcd :: Int -> Int -> Int
gcd a 0 = a
gcd a b
    | a >= b    = gcd b (mod a b)
    | otherwise = gcd b a
```

Example: Largest divisor

- Find the largest divisor of *n*, other than *n* itself
- Strategy: try *n-1*, *n-2*, ... In the worst case, we stop at *1*

```
largestdiv :: Int -> Int
largestdiv n = divsearch n (n-1)

divsearch :: Int -> Int -> Int
divsearch m i
   | (mod m i) == 0 = i
   | otherwise      = divsearch m (i-1)
```

Example: length of an integer

- Length of *n* is *j* if
  - Dividing *n* by *10 j-1* times gives a number *>= 1*
  - Dividing *n* by *10 j* times takes us below *1*

Integer length ...

- `intlength 256` = `3` because $256/10^2 > 1$, $256/10^3 < 1$

- If $n < 10$ then `intlength n = 1`

- ```
  intlength :: Int -> Int
  intlength n
    | n < 0      = 0
    | n < 10     = 1
    | otherwise = intlength (n `div` 10)
  ```
  ```

Example: Reverse digits

- `intreverse 13276` should yield `67231`
- Strategy
  - Split `13276` as `1327` and `6` using `div` and `mod`
  - Recursively reverse `1327` to get `7231`
  - Multiply `6` by suitable power of `10` and add:
    `60000+7231 = 67231`
  - Use `intlength` to determine the power of `10`

Reverse digits ...

```
intreverse :: Int -> Int
intreverse n
   | n < 10  = n
   | otherwise = (intreverse (div n 10)) +
                 (mod n 10)*
                     (power 10 ((intlength n)-1))

power :: Int -> Int -> Int
power m 0 = 1
power m n = m * (power m (n-1))
```

Reverse digits ...

```
intRev :: Int -> Int
intRev n = intRev' n 0

intRev' :: Int -> Int -> Int
intRev' 0 m = m
intRev' n m = intRev' (n `div` 10) (10*m + n `mod` 10)
```