

Programming in Haskell

Aug-Nov 2015

LECTURE 3

AUGUST 11, 2015

S P SURESH

CHENNAI MATHEMATICAL INSTITUTE

Example: gcd

- * Euclid's algorithm for $\text{gcd}(a,b)$, assume $a \geq b$
 - * If b is 0, define the answer to be a
 - * Otherwise, $\text{gcd}(a,b) = \text{gcd}(b, \text{mod } a \ b)$

```
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 $n-1, n-2, \dots$. 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 `length n = 1`
- * `intlength :: Int -> Int`
`intlength n`
 - | $n < 0$ = `intlength n (n `div` 10)`
 - | $n < 10$ = `1`
 - | otherwise = `0`

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 length 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)
```