(+), (*), (-) :: Num a => a -> a -> a
(/) :: Fractional a => a -> a -> a
div, mod :: Integral a => a -> a -> a
13 'div' 5 = 2 13 'mod' 5 = 3 13 / 5 = 2.6

(^) :: (Num a, Integral b) => a -> b -> a
even, odd :: Integral a => a -> Bool

(<), (<=), (>), (>=) :: Ord a => a -> a -> Bool
(==), (/=) :: Eq a => a -> a -> Bool
(&&), (||) :: Bool -> Bool -> Bool
not :: Bool -> Bool
max, min :: Ord a => a -> a -> a
max 3 7 = 7 min 3 7 = 3

round :: (RealFrac a, Integral b) => a -> b
fromIntegral :: (Integral a, Num b) => a -> b
round 2.3 = 2 fromIntegral(length [1,2]) + 3.2 = 5.2

To use the following functions: import Data.Char

isAlpha, isLower, isUpper, isDigit :: Char -> Bool
isAlpha 'a' = True isAlpha '3' = False
isLower 'a' = True isLower 'A' = False

toLower, toUpper :: Char -> Char
toLower 'A' = 'a' toUpper 'a' = 'A'

digitToInt :: Char -> Int
intToDigit :: Int -> Char
digitToInt '3' = 3 intToDigit 3 = '3'

Figure 1: Some functions on basic data
sum, product :: (Num a) => [a] -> a and, or :: [Bool] -> Bool
sum [1,0,2,0,3,0] = 6.0 and [True,False,True] = False
product [1,2,3,4] = 24 or [True,False,True] = True

maximum, minimum :: (Ord a) => [a] -> a reverse :: [a] -> [a]
maximum [3,1,4,2] = 4 reverse "goodbye" = "eybdoog"
minimum [3,1,4,2] = 1

concat :: [[a]] -> [a] (++) :: [a] -> [a] -> [a]
concat ["go","od","bye"] = "goodbye" "good" ++ "bye" = "goodbye"

(!!) :: [a] -> Int -> a length :: [a] -> Int
[9,7,5] !! 1 = 7 length [9,7,5] = 3

head :: [a] -> a tail :: [a] -> [a]
head "goodbye" = 'g'
tail "goodbye" = "oodbye"

init :: [a] -> [a] last :: [a] -> a
init "goodbye" = "goodby" last "goodbye" = 'e'

takeWhile :: (a->Bool) -> [a] -> [a] take :: Int -> [a] -> [a]
takeWhile isLower "goodBye" = "good" take 4 "goodbye" = "good"

dropWhile :: (a->Bool) -> [a] -> [a] drop :: Int -> [a] -> [a]
dropWhile isLower "goodBye" = "Bye" drop 4 "goodbye" = "bye"

elem :: (Eq a) => a -> [a] -> Bool replicate :: Int -> a -> [a]
elem 'd' "goodbye" = True replicate 5 '*' = "*****"

zip :: [a] -> [b] -> [(a,b)]
zip [1,2,3,4] [1,4,9] = [(1,1),(2,4),(3,9)]

To use the following function: import Data.List

isPrefixOf :: Eq a => [a] -> [a] -> Bool
isPrefixOf "abc" "abcde" = True

Figure 2: Some functions on lists