Task 1

Write a function that removes the digit d from the number n.

Examples:

removeD 5 656 -> 66

removeD 6 656 -> 5

removeD 0 606 -> 66

removeD 0 600 -> 6

removeD 1 656 -> 656

reverseDigit :: Int -> Int

reverseDigit number

| number < 10 = number

| otherwise = helper number 0

where

helper :: Int -> Int -> Int

helper number newNumber

| number < 10 = newNumber \* 10 + number

| otherwise = helper (div number 10) (newNumber \* 10 + (mod number 10))

removeDigit :: Int -> Int -> Int

removeDigit d n

| n < 10 = if n == d then 0 else n

| otherwise = reverseDigit(helper d n 0)

where

helper :: Int -> Int -> Int -> Int

helper digit number newNumber

| number < 10 = if number == digit then newNumber else newNumber \* 10 + number

| mod number 10 == digit = helper digit (div number 10) newNumber

| otherwise = helper digit (div number 10) (newNumber \* 10 + (mod number 10))

Task 2

Given a divisor d and a bound b, find the largest integer N, such that:

N is divisible by divisor

N is less than or equal to bound

N is greater than 0.

Examples:

maxMultiple 2 7 -> 6

maxMultiple 3 10 -> 9

maxMultiple 7 17 -> 14

maxMultiple 10 50 -> 50

maxMultiple 37 200 -> 185

divison :: Int -> Int -> Int

divison divider number

| number < 0 = error "Can't work with negative numbers"

| mod number divider == 0 = number

| otherwise = divison divider (number - 1)

Task 3

Write a function that finds the maximum digit in a number by searching from right to left.

maxD :: Int -> Int

maxD number

| number < 10 = number

| otherwise = helper (div number 10) (mod number 10)

where

helper :: Int -> Int -> Int

helper leftOver currentMax

| leftOver < 10 = if leftOver > currentMax then leftOver else currentMax

| mod leftOver 10 > currentMax = helper (div leftOver 10) (mod leftOver 10)

| otherwise = helper (div leftOver 10) currentMax

Task4

Write a function that finds the number of elements in a list.

elem :: (Foldable t, Eq a) => a -> t a -> Bool infix 4

Does the element occur in the structure?

myLength [0, 1, 2] -> 3

Task 5

Write a function that generates a list made up of the numbers in the interval (a, b).

inBetween 1 9 -> [2, 3, 4, 5, 6, 7, 8]

Task 6

Write a function that removes every element equal to x in a list.

removeAll 5 [5] -> []

removeAll 4 [4, 4] -> []

removeAll 5 [1] -> [1]

removeAll 5 [5, 1, 5, 3, 5] -> [1, 3]

removeAll 3 [5, 1, 5, 3, 5] -> [5, 1, 5, 5]

removeX :: [Int] -> Int -> [Int]

removeX lst number = helper [] lst number

where

helper :: [Int] -> [Int] -> Int -> [Int]

helper newList [] number = newList

helper newList (a:as) number

| a == number = helper newList as number

| otherwise = helper (a : newList) as number

Task 7

Write a function incrementBy that takes a list of integers, an integer and returns a new list made up of the integers from the first argument but incremented by the second argument.

incrementBy 5 [5] -> [10]

incrementBy 4 [4, 4] -> [8, 8]

incrementBy 5 [1] -> [6]

incrementBy 5 [5, 1, 5, 3, 5] -> [10, 6, 10, 8, 10]

incrementBy 3 [5, 1, 5, 3, 5] -> [8, 4, 8, 6, 8]

incrementBy :: Int -> [Int] -> [Int]

incrementBy n ls = [ x + n | x <- ls]

incrementBy2 n ls = helper [] n ls

where

helper :: [Int] -> Int -> [Int] -> [Int]

helper newList number [] = reverse newList

helper newList number (a:as) = helper ( (a + number) : newList) number as