**1.filter function**  
high \_ [] = []

high f (x:xs) = f x : high f xs

**2.simple interest**

simple :: Double->Double->Double->Double

simple p r t = (p\*r\*t)/100

**3.factors of a number**

fact::Int->[Int]

fact 0 = []

fact n = [y | y <- [1..n], (n `mod` y) == 0]

**4.cm to m**

convert::Double->Double

convert cm = (cm/100)

**5.lower case letters in a string**

cntLo::String->Int

cntLo s = length [y | y <- s, (y `elem` ['a'..'z']) == True ]

**6.making elements of list negative**

neg::[Int]->[Int]

neg [] = []

neg (x:xs)

|x<0 = (-x) : neg xs

|otherwise = x : neg xs

**7.greater than (given 2 nums) using if else**

gt::Int->Int->Int

gt a b = if a>b then a else b

**8.greater than using guard**

gt::Int->Int->Int

gt a b

|a>b = a

|otherwise = b

**9.grade from mark**  
grade::Int->Char

grade a

|a > 80 = a

|a > 60 = b

|otherwise = c

**10.filter**

ab::(Int->Int)->[Int]->[Int]

ab \_ [] = []

ab f (x:xs) = f x : ab f xs

**11. <15**

ml::[Int]->[Int]

ml x = [2\*y | y <- x, 2\*y < 15]

**12.map function**

ma :: (Int->Int)->[Int]->[Int]

ma \_ [] = []

ma f (x:xs) = f x : ma f xs

**13.root**

root a b = (k, m) where

k = a+b

m = a-b

**14. returns 3**

cy a b = let s = 2

t = 1

in s+t

**15. Returns kamal for 1 and hello for 2**

cy a = case a of

1 -> "PPL"

2 -> "Hello"

**16. zip**

lit::(Int->Int->Int)->[(Int, Int)]->[Int]

lit \_ [] = []

lit f (x:xs) = f (fst x) (snd x) : lit f xs

**17. f to c**

fn :: Double->Double

fn c = (c\*(9/5)) + 32

**18. sum of elements using foldL**

fn :: [Int]->Int

fn xs = foldl (\acc x -> acc +x ) 0 xs

-}

fn :: [Int]->Int

fn xs = foldl1 (\acc x -> acc \*x ) xs