POPL (SET-2)

CB.EN.U4CSE22544

SHREYA.J.V

6)

myFilterFunction :: (Int -> Bool) -> (Int -> Int) -> [Int] -> [Int]

myFilterFunction predicate func = map (\x -> if predicate x then func x else x)

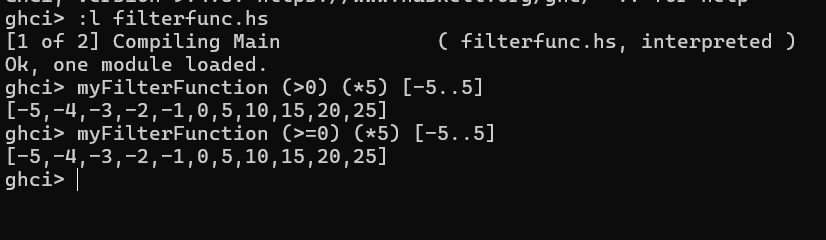
ghci:

ghci> myFilterFunction (>0) (\*5) [-5..5]

[-5,-4,-3,-2,-1,0,5,10,15,20,25]

ghci> myFilterFunction (>=0) (\*5) [-5..5]

[-5,-4,-3,-2,-1,0,5,10,15,20,25]



7)

retainDigits :: [String] -> [String]

retainDigits = map filterDigits

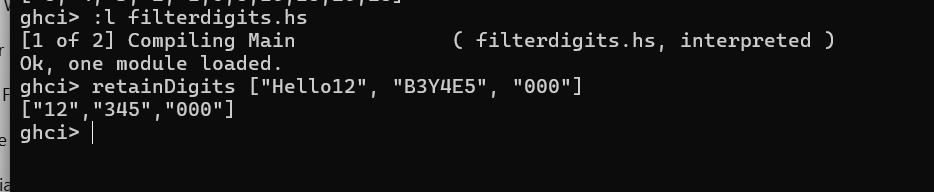
where

filterDigits [] = [] -- Base case: empty string returns an empty string

filterDigits (x:xs)

| x >= '0' && x <= '9' = x : filterDigits xs -- Keep digit characters

| otherwise = filterDigits xs -- Skip non-digit characters



8)

sortedmerge :: (Ord a) => [a] -> [a] -> [a]

sortedmerge xs [] = xs

sortedmerge [] ys = ys

sortedmerge (x:a) (y:b)

| x < y = x : sortedmerge a (y:b)

| x == y = y : sortedmerge a b

| x > y = y : sortedmerge (x:a) b

