```haskell
-- Laboratorul 5

-- 1. Calculati suma pătratelor elementelor impare dintr-o listă dată ca
-- parametru.

sumPatrate :: [Int] -> Int
sumPatrate lista = sum (map (\x -> x*x) (filter odd lista))

-- 2. Scrieti o functie care verifică faptul că toate elementele dintr-o listă --
-- sunt True, folosind foldr.

elemTrue :: [Bool] -> Bool
elemTrue = foldr (&&) True

-- 3. Scrieti o functie care verifică dacă toate elementele dintr-o listă de
-- numere întregi satisfac o proprietate dată ca parametru.

allVerifies :: (Int -> Bool) -> [Int] -> Bool
allVerifies f lista = foldr (&&) True (map (\h -> f h) lista)

-- 4. Scriet o functie care verifică dacă există elemente într-o listă de numere
-- întregi care satisfac o proprietate dată ca parametru.

anyVerifies :: (Int -> Bool) -> [Int] -> Bool
anyVerifies f lista = foldr (||) False (map (\h -> f h) lista)

-- 5. Redefiniti functiile map si filter folosind foldr. Le puteti numi mapFoldr
-- si filterFoldr.

mapFoldr :: (a -> b) -> [a] -> [b]
mapFoldr f lista = foldr (\ h t -> f h : t) [] lista

filterFoldr :: (a -> Bool) -> [a] -> [a]
filterFoldr f = foldr(\ h t -> if f h then h : t else t) []

-- 6. Folosind functia foldl, definiti functia listToInt care transformă o lista
-- de cifre (un număr foarte mare stocat sub formă de listă) în numărul intreg
-- asociat. Se presupune ca lista de intrare este dată corect.

listToInt :: [Integer] -> Integer
listToInt  = foldl (\ x y -> x *10 + y) 0

-- 7.
-- (a) Scrieti o functie care elimină un caracter din sir de caractere.
```

```haskell
rmChar :: Char -> String -> String
rmChar chr = filter(/=chr)

-- (b) Scrieti o functie recursivă care elimină toate caracterele din al doilea
-- argument care se găsesc în primul argument, folosind rmChar.

rmCharsRec :: String -> String -> String
rmCharsRec [] str = str
rmCharsRec (h:t) str = rmCharsRec t (rmChar h str)

-- (c) Scrieti o functie echivalentă cu cea de la (b) care foloseste foldr în
-- locul recursiei si rmChar.

rmCharsFold :: String -> String -> String
rmCharsFold str = foldr (\ h t -> if h `elem` str then t else h : t) []
```