Zadatak 1. Implementirajte funkciju *dotProd* koja računa skalarni produkt dva vektora iste duljine.

Zadatak 2. Implementirajte funkciju *len* koja vraća duljinu liste.

Zadatak 3. Implementirajte funkciju *prodWithScalar* koja vektor množi skalarom.

Zadatak 4. Implementirajte funkciju *vecSum* koja zbraja dva vektora iste duljine.

Zadatak 5. Implementirajte funkciju *countA* koja broji broj pojavljivanja znaka 'a' u nizu znakova.

Zadatak 6. Implementirajte funkciju *countC* koja broji broj pojavljivanja proizvoljnog znaka u nizu znakova.

Zadatak 7. Promotrite sljedeće funkcije.

```
head :: [a] -> a
head [] = error "Lista je prazna."
head (x:xs) = x
```

```
tail :: [a] -> [a]
tail [] = error "Lista je prazna"
tail (x:xs) = xs
```

```
(!!) :: [a] -> Int -> a
(!!) [] _ = error "Lista je prazna"
(!!) _ 0 = error "Ne postoji 0-ti element liste"
(!!) (x:xs) 1 = x
(!!) (x:xs) n = (!!) xs (n-1)
```

```
take :: Int -> [a] -> [a]
take 0 _ = []
take _ [] = error "Lista je prazna"
take n (x:xs) = x:(take (n-1) xs)
```

```
drop :: Int -> [a] -> [a]
drop 0 xs = xs
drop _ [] = error "Lista je prazna"
drop n (x:xs) = drop (n-1) xs
```

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs
```

```
sum :: [Int] -> Int
sum [] = 0
sum (x:xs) = x + sum xs
```

```
product :: [Int] -> Int
product [] = 1
product (x:xs) = x * product xs
```

```
(++)  :: [a] -> [a] -> [a]
(++) []     ys = ys
(++) (x:xs) ys = x:((++) xs ys)
```

```
reverse :: [a] -> [a]
reverse [] = []
reverse (x:xs) = (reverse xs) ++ [x]
```

Napomena: Sve funkcije navedene u ovom zadatku nalaze se u Haskellovoj standardnoj biblioteci.

Zadatak 8. Testirajte funkcije iz *dotProd* i *vecSum* sa argumentima koji su vektori različitih duljina. Po uzoru na primjere iz prethodnog zadatka dopunite te funkcije tako da ispišu poruku o greški ukoliko su im argumenti vektori različitih duljina.