List, More Basic Practice

Implement the functions below by structural recursion over lists.

Recall that the syntax of pattern-matching on a list is as follows (where `x` is the head of the list and `xs` is the tail):

```
function :: [...] -> ...
function []     = ...
function (x:xs) = ...
```

1. Write a function `delete5 :: [Integer] -> Integer`, which takes a list of integers and removes all occurrences of 5.

```
Main> delete5 [1,2,3]
[1,2,3]

Main> delete5 [1,2,3,4,5]
[1,2,3,4]

Main> delete5 [5,1,5,7]
[1,7]
```

2. Write a function `dropAlternating :: Bool -> [a] -> [a]`, which drops alternating elements from the list. The boolean parameter indicates whether the first element should be dropped.

```
Main> dropAlternating True [1,2,3,4]
[2,4]

Main> dropAlternating False [1,2,3,4]
[1,3]

Main> dropAlternating True []
[]

Main> dropAlternating False []
[]
```

3. Write a function `accumulatedSum :: Integer -> [Integer] -> Integer`, which returns the running sum at each element in the list, starting from the given value.

```
Main> accumulatedSum 0 [1,2,3]
[0,1,3,6]
```

```
Main> accumulatedSum 5 [1,2,3]
[5,6,8,11]

Main> accumulatedSum 5 []
[5]
```

4. Write a function `commaTrail :: [String] -> String`, which takes a list of strings and concatenates them after trailing each with a comma.

```
Main> commaTrail []
""

Main> commaTrail ["hello"]
"hello,"

Main> commaTrail ["one","two","three"]
"one,two,three,"
```

5. Write a function `trailBy :: String -> [String] -> String`, which generalizes the previous function from using a comma as separator to using the given string parameter for trailing.

```
Main> trailBy "," ["one","two","three"]
"one,two,three,"

Main> trailBy "\n" ["one","two","three"]
"one\ntwo\nthree\n"

Main> trailBy "" ["one","two","three"]
"onetwothree"
```

6. Write a function `commaSeparate :: [String] -> String`, which takes a list of strings and turns them into a comma-separated string.

```
Main> commaSeparate []
""

Main> commaSeparate ["hello"]
"hello"

Main> commaSeparate ["one","two","three"]
"one,two,three"
```

7. Write a function `separateBy :: String -> [String] -> String`, which generalizes the previous function from using a comma as separator to using the given string parameter as separator.

```
Main> separateBy "," ["one","two","three"]
"one,two,three"

Main> separateBy "-#-" ["one","two","three"]
"one-#-two-#-three"

Main> separateBy "" ["one","two","three"]
"onetwothree"
```