

Accumulate in Haskell

[Readme \(readme\)](#)[Test Suite \(../accumulate\)](#)

Accumulate

Implement the `accumulate` operation, which, given a collection and an operation to perform on each element of the collection, returns a new collection containing the result of applying that operation to each element of the input collection.

For example, given the collection of numbers:

- 1, 2, 3, 4, 5

And the operation:

- square a number

Your code should be able to produce the collection of squares:

- 1, 4, 9, 16, 25

Check out the test suite to see the expected function signature.

Restrictions

Keep your hands off that `collect/map/fmap/whatchamacallit` functionality provided by your standard library!

Solve this one yourself using other basic tools instead.

Elixir specific: it's perfectly fine to use `Enum.reduce` or `Enumerable.reduce`.

Lisp specific: it's perfectly fine to use `MAPCAR` or the equivalent, as this is idiomatic Lisp, not a library function.

Check out Exercism

Help (<http://help.exercism.io/getting-started-with-haskell.html>) for instructions to get started writing Haskell.

Running Tests

Use `runhaskell` (included in the Haskell Platform) to compile and run your Haskell code.

```
1 $ runhaskell -Wall bob_test.hs
```

Source

Conversation with James Edward Gray II [view source \(https://twitter.com/jeg2\)](https://twitter.com/jeg2)

exercism.io

(/)

Beta

[About \(/about\)](/about) - [Donate \(/donate\)](/donate)

 [GitHub \(https://github.com/exercism/exercism.io\)](https://github.com/exercism/exercism.io)  [Twitter \(https://twitter.com/exercism_io\)](https://twitter.com/exercism_io)

 [Newsletter \(https://tinyletter.com/exercism\)](https://tinyletter.com/exercism)

SPONSORS



<https://bugsnag.com/blog/bugsnag-loves-open-source>



<http://www.rackspace.com/>



<http://www.shopify.com/>

© 2015 Katrina Owen