Exercises Types

You can use `:t` in the interpreter to see the type of an expression, however, try to use it only after you've thought about and maybe written down the type yourself!

1 What is the type?

Try to write the type of the following values

- `['a', 'b', 'c']`
- `('a', 'b', 'c')`
- `(True, 'a')`
- `[(True, 'a'), (False, 'b')]`
- `([True, False], ['a','b'])`
- `tail`
- `reverse`
- `[tail, reverse]`

2 What was the type?

Try looking and some of the following functions that we have used previously, and understand their types. Make your guess first, then check it with `:t`.

- `length`
- `head`
- `null`
- `take`
- `maximum`
- `sum`

- `elem`
- `repeat`
- `cycle`
- `succ`

3 Make the function

Write the type for the following functions, then implement them. You are welcome to use functions you have seen or made previously (this does not include `:t`).

- `second`, takes a list and returns the second element.
- `swap`, takes a 2-tuple and returns a tuple with the elements in reverse order.
- `pair`, takes two elements and returns a tuple of them.
- `double`, takes a number and returns the double of it.
- `palindrome`, takes a list and returns whether it is the same forwards and backwards. (Hint: use `reverse` in the implementation)
- `twice`, takes a function and an element the function can be applied to, and applies it twice.