Lab 10: More on Maybes

This week, we'll get even more acquainted with the `Maybe` type constructor.

Starter code

- `Lab10.hs`

Task 1: More practice with `Maybe`

Implement the following functions according to their type signatures. Note that each of the first four has only one "reasonable" implementation that can typecheck. Part of your thought process here should be to determine what each function does, and explain it briefly *in English*!

1. `mapMaybes :: (a -> b) -> [Maybe a] -> [Maybe b]`
2. `composeMaybe :: (a -> Maybe b) -> (b -> Maybe c) -> (a -> Maybe c)`
3. `foldMaybe :: (b -> a -> Maybe b) -> b -> [a] -> Maybe b`
4. `applyBinaryMaybe :: (a -> b -> c) -> Maybe a -> Maybe b -> Maybe c`.
5. `collectMaybes :: [Maybe a] -> Maybe [a]`. Return a `Just` if the original list contains *only* `Just` values, and no `Nothing`s. This is probably the hardest one, and we strongly recommend using `applyBinaryMaybe` and/or writing helper function(s) with good type signatures here!