Part I catchup

Setup the environment

Every function mentioned as useful below is defined and described here:

http://package.elm-lang.org/packages/elm-lang/core/latest

Note on simplified reading of definitions:

a -> b takes a returns b

a -> b -> c takes a, takes b, returns c – if supplied only first argument, will return b -> c which is another function taking b, returning c

(a -> b) -> c -> d takes function a -> b, takes c, returns d

If you mess something up, compiler will give you a friendly message on what exactly is wrong ☺

Go through:

https://guide.elm-lang.org/core_language.html

https://guide.elm-lang.org/architecture/

https://guide.elm-lang.org/architecture/user_input/buttons.html
=> this is pretty much Ex1

Read https://guide.elm-lang.org/reuse/modules.html (without Building Projects with Multiple Modules)

Ex2:

- Type in:

```
square n =
  n^2

hypotenuse a b =
  sqrt (square a + square b)

distance (a,b) (x,y) =
  hypotenuse (a-x) (b-y)
```

- Open elm-repl
- Import Ex2 module
- Run those functions (square 5, etc.)
- Type in:

```
squareAnon =
  \n -> n^2

squares =
  List.map (\n -> n^2) (List.range 1 100)
```

- Run
- Implement function add5 which always returns value incremented by 5
- Play around with basic operations

Ex3:

Use elm-reactor to open Ex3

Implement Ceasar's Cypher

1. Let expression http://elm-lang.org/docs/syntax#let-expressions
2. Function application operators:
   a. http://package.elm-lang.org/packages/elm-lang/core/latest/Basics#|>
   b. http://package.elm-lang.org/packages/elm-lang/core/latest/Basics#<|
3. Conditionals http://elm-lang.org/docs/syntax#conditionals
   ---
4. Implement function shiftedIndex which for every list returns list of same length with integer indexes, shifted by given key, e.g.

i. **['A', 'B', 'C'] 0 -> [0, 1, 2]**
ii. **['A', 'B', 'C'] 1 -> [1, 2, 0];**

b. Useful functions:

```
List.length : List a -> Int
List.range : Int -> Int -> List Int
List.map : (a -> b) -> List a -> List b
```

5. Uncomment lines 21 and 22, implement functions

```
indexList values key =
  ???

appendIndex values key =
  ???
```

Where both take a list and:

- **indexList** returns list of tuples with first element being shifted index and second original item, e.g.
  - ['A', 'B', 'C'] 1 => [(1, 'A'), (2, 'B'), (0, 'C')]
- **appendIndex** returns list of tuple with first element being original item and second being it's index, e.g.
  - ['A', 'B', 'C'] 1 => [('A', 1), ('B', 2), ('C', 0)]

Another useful function:

List.map2 : (a -> b -> result) -> List a -> List b -> List result

6. Read https://guide.elm-lang.org/error_handling/maybe.html, pay attention to 'case' construction
7. Lookup definitions of those functions, try using them in elm-repl:

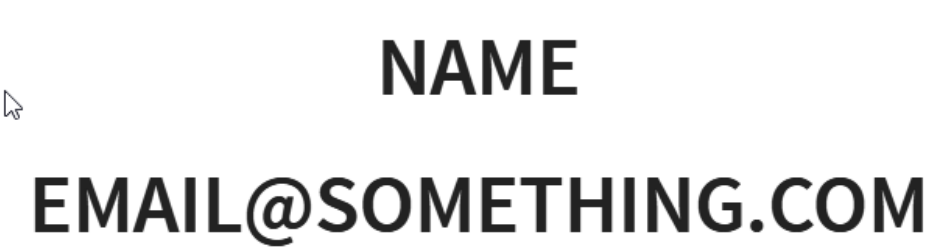
```
Dict.fromList : List (comparable, v) -> Dict comparable v
Dict.get : comparable -> Dict comparable v -> Maybe v
Maybe.withDefault : a -> Maybe a -> a
String.map : (Char -> Char) -> String -> String
```

8. Uncomment line 23

9. Implement
```
encode message key =
  ???
```
    a. Hint: consider creating two dictionaries which will lookup values from char to int and from int to char.
    b. Hint-2: You can import Ex3 module from elm-repl and experiment with functions without writing any view code
    c. You can ‘get inspired’ by solution in Solutions/Ex3

Ex4

1. Read https://guide.elm-lang.org/types/type_aliases.html
2. Read https://guide.elm-lang.org/types/union_types.html
3. Run elm-reactor, open Ex4 and get it to state where original user’s data is shown as two h2 elements:



4. Hint: all html nodes are represented as functions which take a list of attributes (which are also functions) and list of content nodes, e.g.:
   div [style [("width", "400px")]] [text “ABC”]

Ex5

Use knowledge about types from Ex4 to show new user’s data along with his notes:

Dan

dmaterowski@infusion.com

- **Header**

  And some content for the sake of taking up space.
  And even more lines, and stuff and like you know,
  something meaningful. 1

-  2

- **I like trains**

  Really!!! 3