Writing Functions on a File

1. Open any text editor (notepad, textedit, vim, pico, etc.).

2. Type the following:

```
my_function x = head (reverse x)
```

3. Save the file, name it as **csci2100_2_17_22.hs**

   [Note: .hs extension tells the computer that this is a source code/file written in Haskell]

4. Keep the file in a particular directory/location. For example, I am putting the file in the following folder/directory:

   `/Users/fhamid/Desktop/CSCI2100Code`

5. Load GHCi from the same directory. Once you see **Prelude>**, type the following command:

   Prelude> :l csci2100_2_17_22.hs

   ■ Treat `l` as an acronym of `load`.
   You will see some information like the following:

```
[1 of 1] Compiling Main             ( csci2100_2_17_22.hs, interpreted
)
      Ok, one module loaded.
     *Main>
```

   It means that the program has been loaded successfully by the compiler.

6. Type **:browse** on the prompt and see what happens. You may see the following:

   my_function :: [a] -> a

   Now, we probably understand what this means, right? This means -

   *my_function takes a list of any type **a** as input and returns a single value (of type **a**) as output.*

7. Now it is time to test the function. Try at least the following samples:

```
*Main> my_function [1..10]
10

*Main> my_function "apple"
'e'

*Main> my_function ["apple", "orange", "grapes"]
"grapes"

*Main>
```

8. Open the source file (**csci2100_2_17_22.hs**) and add the following line. Save the file.

```
my_function :: Fractional a => [a] -> a

my_function x = head (reverse x) --definition
```

- Read it like "*my_function takes a list of **Fractoional** values as input and returns a single Fractional value as output.*"

9. Go to the Haskell prompt. If you see **"*Main>"** then type **:r** (reload) or **:reload**. This will reload the file with the changes you have made. You may also type - **:l csci2100_2_17_22.hs**

10. Test the function again with the same sets as step 7. What changes do you observe? Write your observation as

11. Go to the same source file and add the definition of another function (squareXY) after my_function.

```
squareXY x y = x * x + y * y
```

12. Type **:browse** on the main prompt. Carefully check what is shown.
13. Now test squareXY with the following input sets:

```
*Main> squareXY 3.5 1
???
*Main> squareXY 3.5 (-2.5)
???
*Main> squareXY 3 (-2)
???
*Main> squareXY 3 2
???
```

14. Add type restrictions so that `squareXY` can only handle Integrals. Reload the file and test. In your test cases, add at least one Fractional sample. Copy and paste your output as multiline comments in the source file.

    [Hint: multi-line comment in Haskell:
    https://www.haskell.org/haddock/doc/html/markup.html]

15. Write a function `play_tuples` that produces the following output:

```
*Main> play_tuples "Hello" "2100"
('H','2')
*Main> play_tuples "Hello" [1..10]
('H',1)
*Main> play_tuples [1..10] [5,5,5]
(1,5)
*Main> play_tuples ['c', 'a', 't'] [5.0,5.0,5.0]
('c',5.0)
```

16. Test the type of your defined function play_tuples. Interpret the output of the compiler. Now add proper commands/instructions to make sure that it shows similar output as the following:

```
*Main> play_tuples ['c', 'a', 't'] [5.0,5.0,5.0]

<interactive>:61:30: error:
    • No instance for (Fractional Char) arising from the literal ‘5.0’
    • In the expression: 5.0
      In the second argument of ‘play_tuples’, namely ‘[5.0, 5.0, 5.0]’
      In the expression: play_tuples ['c', 'a', 't'] [5.0, 5.0, 5.0]
*Main> play_tuples ['c', 'a', 't'] ['g', 'o']
('c','g')
*Main> play_tuples "cat" "dog"
('c','d')
*Main>
```

Submission:

Submit `csci2100_2_17_22.hs` along with the **worksheet_2_17_2022.(pdf/txt/doc)** file.

- **Don't forget to include your name at the beginning of both of the files.**
- **For the .hs file, your name must be added as a comment.**