COMP105: Programming Paradigms
Week 2 Homework Sheet

This is the homework sheet for **Week 2**. Complete your answers in a file named `week2.hs` and submit them to the "Week 2" assessment in SAM here

`https://sam.csc.liv.ac.uk/COMP/Submissions.pl`

Submission of the weekly homework sheets contributes 10% of the overall module mark, and each homework sheet counts equally towards this. Each homework sheet will be marked on a pass/fail basis. You will receive full marks for submitting a *reasonable attempt* at the homework. If no submission is made, or if a non-reasonable attempt is submitted, then no marks will be awarded.

The deadline for submission is

Friday Week 2 (23/10/2020) at 16:00.

Late submission is **not** possible. Individual feedback will not be given, but full solutions will be posted promptly after the deadline has passed.

If you feel that you are struggling with the homework, or if you have any other questions, then you can contact the lecturer at any point during the week via email, or you can drop in to the weekly Q&A session on MS Teams on Friday between 1PM and 4PM.

Lecture 4 - Ifs. Implement the following functions:

1. Implement a function `gt_100` with one argument `x`, which returns 1 if $x > 100$, and 0 otherwise.
2. Implement a function `switch` with three arguments `x`, `y`, and `c`. If c is equal to 1 then the function should return x, otherwise it should return y.
3. Implement a function `fizzbuzz` that takes one argument `x` and returns "Fizzbuzz!" if x modulo 3 is 0 and x modulo 5 is 0. Otherwise it should return "Nope".

Lecture 4 - Let expressions. Use the `let` syntax from Lecture 4 to implement the following functions. Try out both the single-line and multi-line versions of `let`, and make sure to remember Haskell's layout rule.

1. Write a function `question1 x` that sets $a = x * x$ and returns $2 * a$.
2. Write a function `question2 x` that sets $a = x + 1$, $b = a * a$, and $c = 2^b$, and then returns $a + b - c$.
3. Write a function `bounded_square x` that returns x*x if x*x is less than 100, and 100 otherwise. You should use a `let` and an `if`.

Lecture 5 - Tuples.

1. Write a function `square_and_cube x` that returns a two-element tuple, where the first element is x*x, and the second is x*x*x
2. Write a function `add_tuple (a, b)` that takes a tuple with two elements called a and b, and returns $a + b$.
3. Write a function `swap` that takes a two-element tuple, and swaps the order of the elements of that tuple.

Lecture 5 - Lists.

1. Use the `head` function to write a function `head_squared list` that takes a list as an argument, and returns the square of the head of that list.
2. Use the `!!` operator, write a function `third list` that returns the third element of the input list.
3. Using the `head` and `tail` library functions, write a function `third_head list` that returns the third element of the input list.
4. Using the `:` operator, write a function `prepend_two list a b` that takes a list and two other arguments, and returns a new list with a and b added to the front.

Lecture 5 - List functions. These exercises cover the Prelude list functions discussed in Lecture 5.

1. Use the `length` function to write a function `two_lengths list1 list2` that takes two lists, and returns the sum of their lengths.
2. Use the `reverse` function and the `++` operator to write a function `make_palindrome list` that returns the list followed by the reverse of the list.
3. Use the `sum` and `product` functions to write a function `sum_and_product list` that returns a tuple where the first element is the sum of the list, and the second element is the product of the list.
4. Use the `take` and `drop` functions to write a function `four_through_six list` that returns a list containing elements four, five, and six of the input list.
5. Use the `elem` function to write a function `both_in list x y` that returns True if both `x` and `y` are in `list`.

Lecture 6 - List ranges. In GHCI, use a list range to write a query that outputs:

1. The list of all numbers between 101 and 200.
2. The list of all even numbers between 1000 and 1050.
3. The list of all numbers between 20 and 1 counting backwards.

4. An infinite list of all numbers divisible by 3 starting from 999. Press control+c to stop the print out.

In your file, put each of the list ranges in a comment, like so.

```
-- 6.1 = ...
-- 6.2 = ...
-- etc
```

Lecture 6 - List comprehensions.

1. In GHCI, use the `^` operator to write a list comprehension that outputs the first ten powers of two. Copy the list range into your file in a comment.
2. Write a function `only_odds list` that returns only the odd elements of the input list.
3. Write a function `between a b list` that takes two numbers $a < b$, and returns the elements of `list` that are (strictly) between a and b.
4. (*) Write a function `number_of_es string` that returns the number of times that `'e'` occurs in the input string.
5. (**) Write a function `proper_fizzbuzz` that returns an infinite list with the following properties. In position i of the list,
   - if i is divisible by 3 then the list should contain `"fizz"`
   - if i is divisible by 5 then the list should contain `"buzz"`
   - if i is divisible by both 3 and 5 then the list should contain `"fizzbuzz"`
   - if i is not divisible by 3 or 5 then the list should contain the number i (the `show` function from Prelude will turn an integer into a string.)