Curriculum X Functional Programming

Sheet 1

Instructions

All these questions should be solved by typing expressions at the terminal prompt within GHCi. For some questions you will need to use functions from the `Data.Char` module. To use this module, type `:m +Data.Char` at the prompt. Once you have imported `Data.Char`, your terminal prompt should look similar to this:

```
Prelude Data.Char>
```

To submit your answers, please either type your answers in the Sheet 1 page of the Homework section in your OneNote, or paste screenshots of your terminal. Include both the expression you evaluated and the resulting value. If you choose to use screenshots, it is okay to include multiple questions in the same screenshot. For example:

```
1 (a)   2 - 3 - 4
        -5
```

or

```
Prelude Data.Char> 2 - 3 - 4
-5
```

Grading

To achieve an A on this homework, you should complete question 1, as well as either writing good explanations to any follow on questions and completing one other question, or completing two other questions.

To achieve a G on this homework, you should complete question 1 with good explanations to any follow on questions, as well as an indication of what you learnt if any of your predictions were wrong. You should also answer at least three other questions well.

To achieve a G+ on this homework, you should satisfy all criteria for a G as well as attempting the remaining two questions.

Questions

1. For this question, write down the resulting value you are expecting, then type in the expression and check that you were right. If your prediction was wrong, can you see why it was wrong? If you try again with a similar expression, was your prediction right this time?

   (a) `6 - 4 - 7`

   (b) `6 - (4 - 7)`
   Is subtraction left-associative or right-associative?

   (c) `13 + 47`

   (d) `(+) 13 47`
   What kind of operator is `+`? What kind of operator is `(+)`?

   (e) `30 ‘div‘ 5 ‘div‘ 3`

   (f) `div (div 30 5) 3`
   What kind of operator is `div`? What kind of operator is `‘div‘`?

   > Note that here we are using back ticks (‘) and **not** single quotation marks (’). Back ticks are used to change the way we write an expression and single quotation marks are used to delimit characters.

   (g) `div 40 7`

   (h) `2 ^ 3 ^ 2`
   Is `^` left-associative or right-associative?

   (i) `6 - 3 * 8`

   (j) `2 ^ 3 * 4`

   (k) `2 * 3 ^ 4`
   What can you say about the relative precedence of the `-`, `*` and `^` operators?

> An operator's precedence is used to determine the order in which multiple operator applications are evaluated. In Haskell, precedence is stored as an integer between 0 and 9, where a higher number corresponds to a higher precedence.
> When evaluating an expression `(a op1 b op2 c)`, if `op1` has higher precedence than `op2` then the expression evaluates to `((a op1 b) op2 c)`, otherwise it evaluates to `(a op1 (b op2 c))`.
> It is possible to place brackets only where they are strictly needed, however, when writing code we want it to be as easy as possible for someone else to understand. You should therefore use brackets to improve the clarity of your code and to avoid bugs caused by precedence issues.

(l) `(2 + 7 - 1) == (2 - 1 + 7)`

(m) `ord 'd'`

> You may have noticed in parts `d` and `f`, among others, that brackets are not used to indicate function application. In Python you would expect a function application to look like `function(argument1, argument2)`. In Haskell, the same function application would look like `function argument1 argument2`, since spaces are used to denote function application.
> If one of the arguments is itself a function application, such as in `div 10 ((+) 1 1)`, we do require brackets for reasons that will become apparent later.

(n) `chr (ord 'z' - 1)`

(o) `chr (ord 'T' + (ord 'a' - ord 'A'))`

(p) `'e' /= 'e'`

(q) `'a' < 'c'`

(r) `'m' >= 'g'`

(s) `sqrt 3 ^ 2`

2. Use `div` and `mod` to write expressions to find the values below.

   (a) The last digit of the number 291

   (b) The penultimate digit of the number 457

   (c) The second digit of the number 8236

3. As in question 1, predict the resulting value of the expressions below before evaluating them, then experiment with similar expressions until you are able to write an explanation of how comparison operators work on tuples.

   (a) `(3, 5) < (3, 7)`

   (b) `((2, 7), (5, 9)) > ((1, 12), (3, 1))`

   (c) `('h', 2, (7, 'a')) < ('h', 2, (7, 'z'))`

4. Use `div` and `mod` to write an expression which converts the number of seconds since midnight into a tuple containing the number of hours, minutes and seconds since midnight. Write your answer as a let expression where the number of seconds since midnight is 8473, i.e. `let s = 8473 in <your answer>`.

5. Write an expression which converts a polar coordinate (r, θ) to a cartesian coordinate represented by a pair of floats. Write your answer as a let expression where the polar coordinate is $(1, \pi/4)$. If you are stuck you may want to consider questions 4 and 6 first.

6. This question is about pattern matching. We are going to discuss pattern matching further in the next problem class, but you can start exploring now if you want a head start. Predict the resulting value of the expressions below before evaluating them, then explain the result. If you get an error, why?

   (a) `let (a, b) = (4, 1) in (a, b, 'c')`

   (b) `let (s, t) = (7, 8, 9) in s * t`

   (c) `let (a, (b, x)) = (3, (6, True) in if x then a - b else 0`

   (d) `let something = (7, 2) in let ((x, y), z) = (something, '*') in (y, z)`

   (e) `let j = (True, 12) in (True, j)`

These questions have been adapted from questions in Unassessed Exercises Set 1, part of course 120.1 at Imperial College London, written by Tony Field.