3 Types and polymorphism

1. Define the operations 'and' and 'or' using pattern-matching. (In your definition, you might use other symbols than (||) and (&&) to prevent conflicts in definitions; e.g. (|||) and (&&&).)

2. Define the operations 'and' and 'or' using *conditional expressions* (that is, **if** ... **then** ... **else**...). Note that there is more than one plausible way to do it, but only one correct way because of strictness. Now do the same using *guarded equations*.

3. Define a function *charToNum* that converts a digit character to its numeric equivalent; for example,

   charToNum '3' = 3

   (Hint: You will need the predefined function *ord* :: *Char* → *Int*. To use it, add "**import** *Data.Char*" at the top of your Haskell file.)

4. Define a function *showDate* that takes three integers representing the day, month and year, and returns them formatted as a string (Hint: The (++) operator appends two strings, and the *show* function converts a number to a string). For example:

   showDate 9 10 2023 = `"9 October 2023"`

   If that was too easy, make the day number an ordinal:

   showDate 9 10 2023 = `"9th October 2023"`

5. Continuing the "little game" from the lecture, how many functions of type $(a \to a) \to [a] \to [a]$ can you come up with? And how many of type $(a \to b) \to [a] \to [b]$? How does the answer change between total functions (assuming all inputs are well defined, and requiring outputs to be well defined) and partial functions (allowing for undefined inputs and outputs too)?

6. Similarly for $(a \to a) \to (a \to a) \to (a \to a)$ versus $(b \to c) \to (a \to b) \to (a \to c)$?

7. In the lecture we discussed the *free theorem* of type $[a] \to [a]$: any function h of that type must satisfy $map\ f \circ h = h \circ map\ f$. What is the free theorem of type $[a] \to a$? Of type $[a] \to Maybe\ a$? Of type $[a] \to [b] \to [(a, b)]$?