ML – Cheat sheet

Data types

char (#"x")
string ("x")
int
real
bool
a' (all types)
list

Definitions

val <name> = <newValue>;
e.g.: val x = 25;
fun <name> <param> = <expr>;
e.g.: fun addme (a, b) = a + b;
fun <name> <param> = **fn** <name_HOF> => <expr>;
e.g. fun g a = fn b => a + b;
let <defs> **in** <expr> **end**
e.g. let val x = 1 in x*x end

Control flow

if <expr> **orelse** <expr> **andalso** <expr>
then <expr>
else <expr>

```
case <expr> of
        <value1> => <result1> |
        <value2> => <result2> |
        <value3> => <result3>;
```

Patterns

0 => constant 0 (int)
_ => any given param matches
(a, b, c) => tuple of three elements
[a, b, c] => list of three elements
(x :: rest) => matches non-emtpy list, x is head
(x1 :: x2 :: rest) => matches list ≥1 elements, ...
... x1 is head, x2 element after head

```
fun     fact 0 = 1 |
        fact n = n * fact (n-1);
```

Lists and tuples

List = all same type, >= 2 elements
Tuple = all types possible
() => tuple
[] => emtpy list (= nil)
[1] int list with '1'
1 :: 2 :: 3 :: [] => [1, 2, 3]
hd(<list>) => first element
tl(<list>) => rest elements in list
<list1> @ <list2> => combination

Operators

~ (negation)
+, -, *, /, div (int quotient), mod (int remainder)
^ (string concat)

>, <, <=, >= (for all numbers)
=, <> (not for reals!)

Conversions

explode(<string>) => list of chars
implode(<list of chars> => string
real(<int>) => real
floor(<real>) => rounds down to int
ceil(<real>) => rounds up to int
round(<real>) => rounds to int
ord(<char>) => char to ASCII code
chr(<int>) => ASCII code to char
map <fun> <list> => apply <fun>...
...to each element in <list>
foldr <fun> <starter> <list> => f(x1, f(x2, (...) f(xn, c))
foldl <fun> <starter> <list> => f(xn, f(xn-1, (...) f(x1, c))