CPSC 370 **Haskell Cheat sheet** *UNBC*
Fall 2017

Form of Haskell file

```
module statement
[import statements]
declarations
```

Module statements

1. `module name where`
2. `module name(ids ...) where`

Import statements

1. `import` *`mname`*
2. `import` *`mname`*`(`*`names`*`)`
3. `import` *`oname`* `as` *`mname`*`(`*`names`*`)`
4. `import qualified ...`

Declations

```
name :: type
name = value
```

There is short hand syntax for function declarations.

Function declarations

- ```
  name pat1 = expr1
  name pat2 = expr2 ...
  ```
- ```
  name pat1
     | guard1   = expr11
     | guard2   = expr12
  name pat2  = expr2 ...
  ```
- ```
  name pat1
  | guard1   = expr11
  | guard2   = expr12
  name pat2     = expr2 where
     decls ...
  ```

Names

1. keywords include: `class`, `data`, `do`, `else`, `if`, `import`, `instance`, `in`, `let`, `module`, `newtype`, `of`, `then`, `type`, `where`.
2. are like in Java.
3. "_" is special.
4. case matters.

Operators

1. are made from `#$%&-\^!*+./<=>?@|:`
2. `=`, `=>`, `<-`, `->`, `::`, `:` are reserved.
3. Are infix names for binary curried functions.
4. `(+*) a b` is the same as `a +* b`.
5. `(a +*) b` is the same as `a +* b`.
6. `(+* b) a` is the same as `a +* b`.
7. `a` ‘`f`‘ `b` is the same as `f a b`.
8. Operators that start with ":" are constructors.

Layout

- Outside of `{ ... }` line indentation matters.
- `{... ; ...; ... }` can be used in place of indentation.

Constructs

- `if` *`expr`* `then` *`expr`* `else` *`expr`*; types of then and else must match.
- ```
  case expr of
    pat1 -> value1
    pat2 -> value2
    ...
  ```
- ```
  let
    decl1
    decl2 ...
  in expr
  ```
- ```
  expr where
    decl1
    decl2 ...
  ```

Types

- `Int`, `Integer`, `Float`, `Rational`, `Double`, `[a]`, `Maybe a`, `(a,b)`, `a -> b`, *`constraint`* `=>` *`type`*.
- typically occur after `::`.