The Idris Programming Language: Quick Reference Sheet

General	
&idris	Enters Idris' REPL
&idris <*file*>	Loads a file
–codegen <*lang*>	Generates code in another language (currently supports Js and C)
-o <*exe*>	Generates an executable of the given <exe> name

Interactive commands for atom
<*ctrl-alt-a*> : Add clause
<*ctrl-alt-c*> : Case-split
<*ctrl-alt-l*> : Make-lemma (Lift hole)
<*ctrl-alt-s*> : Proof-search a hole
<*ctrl-alt-r*> : Type-check program
<*ctrl-alt-t*> : Shows type of variable
<*ctrl-alt-d*> : Shows documentation
<*ctrl-alt-enter*> : Spawn the REPL

Interactive commands for vim	
r	Reloads file
l <*file*>	Load file
t	Show type
d	Add clause
c	Case-split
l	Make-lemma
p	Proof-search
h	Show documentation

Data and Types	
Int	Primitive integer
Double	Primitive float
Char	Primitive character
Bool	Primitive boolean
Ptr	Foreign pointer
data	Keyword for declaring a new Type
x : <*Type*>	Declares x to be a <*Type*>
a	Stands for any type. (e.g. x : a -> a)

Functions	
<*a*> : <*x*> -> <*y*>	Syntax for declaring a function <*a*> that takes a Type <*x*> and outputs a Type <*y*>.
a x = y	Syntax for defining the function. Every function has both a declaration and definition.
where	Keyword for defining local values.
using	Keyword to explicitly give implicit values names; used before declaration.
x => <*expr*>	Lambda expressions, using a given *x*.
{name: <*a*>}	Example of explicitly associating the implicit argument *name* to the type <*a*>.

Lists and Pattern matching	
General	Lists are homogenous and are comprised of the head of the list, followed by the tail of the rest of the list.
List <*a*>	Defining the Type of the list of type <*a*>.
_	Matches anything.
`(x::xs)`	Matches a list of at least one length.
`(x::[])`	Matches a list of exactly one length.
`(x::y::xs)`	Matches a list of at least two length.
`[]`	Matches an empty list.
`Nil`	Also matches the empty list.

Formal proofs	
General	Idris provides two styles of proofs, one using tactics and the other using Idris' capabilities to find reflexive predicates.
<*a*> : x=x	Defining a predicate <*a*> for proving the theorem x = x
:elab	Enters the interactive theorem prover using Tactics.
:p	Enters the deprecated interactive mode using the old Tactics.
rewrite	Keyword to rewrite a clause given that it can be rewritten into something equivalent.
Refl	Keyword to signify that the expression is reflexive.