Haskell Operators and other Lexical Notation

`--`	Start of comment line
`{-`	Start of short comment
`-}`	End of short comment
`+`	Add operator
`-`	Subtract/negate operator
`*`	Multiply operator
`/`	Division operator
	Substitution operator, as in `e{f/x}`
`^`, `^^`, `**`	Raise-to-the-power operators
`&&`	And operator
`||`	Or operator
`<`	Less-than operator
`<=`	Less-than-or-equal operator
`==`	Equal operator
`/=`	Not-equal operator
`>=`	Greater-than-or-equal operator
`>`	Greater-than operator
`\`	Lambda operator
`.`	Function composition operator
	Name qualifier
`|`	Guard and `case` specifier
	Separator in list comprehension
	Alternative in data definition (enum type)
`++`	List concatenation operator
`:`	Append-head operator ("cons")
`!!`	Indexing operator
`..`	Range-specifier for lists
`\\`	List-difference operator
`<-`	List comprehension generator
	Single assignment operator in do-constr.
`;`	Definition separator
`->`	Function type-mapping operator.
	Lambda definition operator
	Separator in case construction
`=`	Type- or value-naming operator
`::`	Type specification operator, "has type"
`=>`	Context inheritance from class
`()`	Empty value in IO () type
`>>`	Monad sequencing operator
`>>=`	Monad sequencing operator with value passing
`>@>`	Object composition operator (monads)
`(..)`	Constructor for export operator (postfix)
`[` and `]`	List constructors, "," as separator
`(` and `)`	Tuple constructors, "," as separator
	Infix-to-prefix constructors
` ‘ ` and ` ‘ `	Prefix-to-infix constructors
` ’ ` and ` ’ `	Literal char constructors
`"` and `"`	String constructors
`_`	Wildcard in pattern
`~`	Irrefutable pattern
`!`	Force evaluation (strictness flag)
`@`	"Read As" in pattern matching