Haskell Function Definitions

function	op notation	# param	types	example	description
&&	infix	2	Bool, Bool	x>5 && True	logical AND
not	prefix	1	Bool	not True	negation
succ	prefix	1	enumerable type	succ 'a'	succeeding element
\`div\`	infix	2	Num, Num	5 \`div\` 4	integer division
++	infix	2	List, List	[1, 2] ++ [3, 4]	list concatenation
:	infix	2	Object, List	'a' : ['b', 'c']	construct list
!!	infix	2	List, Num	[1..4] !! 2	list accessing by index
..	infix	1 or 2	enumerable type	[1..4]	creates range between two enums or infinity with one param
<=	infix	2	ordered type	'a' <= 'b'	less than or equal to
<-	infix	2	variable, List	x <- [1..10]	draws variable values from list
|	infix	2	variable, Bool	[x | x <- [3..8]], x < 5]	provides for list comprehension through filtering
head	prefix	1	List	head [1..5]	first element in list
tail	prefix	1	List	tail [1..5]	list without head
init	prefix	1	List	init [1..5]	list without tail
last	prefix	1	List	last [1..5]	last element in list
length	prefix	1	List	length [1..5]	length of a list
reverse	prefix	1	List	reverse [1..5]	elements in reversed order
take	prefix	2	Num, List	take 3 ['a'..]	first (number of elements) from list
drop	prefix	2	Num, List	drop 3 ['a'..'r']	remove first (number of elements) from list
elem	prefix	2	Object, List	elem 5 [1..6]	if element is in list
\`elem\`	infix	2	Object, List	5 \`elem\` ['a'..'r']	if element is in list
sum	prefix	1	List	sum [1..6]	sum of all elements in list
fst	prefix	1	Tuple	fst (1, ”square”)	first element in tuple
snd	prefix	1	Tuple	snd (1, “square”)	second element in tuple
zip	prefix	2	List, List	zip ['a'..] [1..5]	list of tuples from both lists