The Original Grammar:

exp → exp || term | term

term → term && factor | factor

factor → factor comop operand | operand

comop → > | = | <

operand → ! operand | Identifier

-We notice that there is left recursion needs to be solved in the following:

exp → exp || term | term

term → term && factor | factor

factor → factor comop operand | operand

Like applying the rule to: A → α A | β

Solution:

A → β A'

A' → α A' | ε

Then:

exp → term exp'

exp' → || term exp' | ε

---

term → factor term'

term' → && factor term' | ε

---

factor → operand factor'

factor' → comp operand factor' | ε

---

comp → > | = | <

operand → ! operand | ID

Production	First	Follow
exp → term exp'	{ ! , ID }	{ $ }
exp' → || term exp' | ε	{ || , ε }	{ $ }
term → factor term'	{ ! , ID }	{ || , $ }
term' → && factor term' | ε	{ && , ε }	{ || , $ }
factor → operand factor'	{ ! , ID }	{ && , || , $ }
factor' → comp operand factor' | ε	{ > , = , < , ε }	{ && , || , $ }
comp → > | = | <	{ > , = , < }	{ ! , ID }
operand → ! operand | ID	{ ! , ID }	{ > , = , < , && , || , $ }

Non Terminal	Input Symbols							
	!	ID	||	&&	>	<	=	$
exp	exp → term exp'	exp → term exp'						
exp'			exp' → || term exp'					exp' → ε
term	term → factor term'	term → factor term'						
term'			term' → ε	term' → && factor term'				term' → ε
factor	factor → operand factor'	factor → operand factor'						
factor'			factor' → ε	factor' → ε	factor' → comp operand factor'	factor' → comp operand factor'	factor' → comp operand factor'	factor' → ε
comp					comp → >	comp → <	comp → =	
operand	operand → ! operand	operand → ID						