

Concepts of Programming Languages, CSCI 305, Fall 2021
Homework #6, complete by Oct. 22

1. Create a grammar for a predicate logic language to generate strings such:

$$\forall x (A(x) \rightarrow (B(y) \leftrightarrow \neg C(z)) \vee D(w))$$

Your language should allow bracketing (left and right parentheses), the logical binary operators: and (\wedge), or (\vee), conditional (\rightarrow), and biconditional (\leftrightarrow). It should also allow the unary operators: not (\neg), universal quantification (\forall), and existential quantification (\exists). Upper case letters of the alphabet denote predicates. Lower case letters of the alphabet denote variables. Predicates only apply to a single variable.

The order of precedence for your language should be:

$$\begin{aligned} &\neg, \forall, \exists \text{ (highest level)} \\ &\wedge \\ &\vee \\ &\rightarrow \\ &\leftrightarrow \text{ (lowest)} \end{aligned}$$

Your grammar should use left associativity so that the predicate

$$A(x) \wedge B(x) \wedge B(x)$$

would be interpreted as

$$((A(x) \wedge B(x)) \wedge B(x)).$$

In order to avoid confusing the predicate logic conditional operator, \rightarrow , with the symbol used to define a grammar, use the symbol \Rightarrow to define your grammar.

2. Show all pushes and pops which occur to a stack when parsing the code,
write (A*B) \$\$

given the following productions and parse table.

1. program \rightarrow stmt_list \$\$
2. stmt_list \rightarrow stmt stmt_list
3. stmt_list \rightarrow ϵ
4. stmt \rightarrow id := expr
5. stmt \rightarrow read id
6. stmt \rightarrow write expr
7. expr \rightarrow term term_tail
8. term_tail \rightarrow add_op term term_tail
9. term_tail \rightarrow ϵ
10. term \rightarrow factor factor_tail
11. factor_tail \rightarrow mult_op factor factor_tail
12. factor_tail \rightarrow ϵ
13. factor \rightarrow (expr)
14. factor \rightarrow id
15. factor \rightarrow number
16. add_op \rightarrow +
17. add_op \rightarrow -
18. mult_op \rightarrow *
19. mult_op \rightarrow /

	id	number	read	write	:=	()	+	-	*	/	\$
program	1	-	1	1	-	-	-	-	-	-	-	1
stmt_list	2	-	2	2	-	-	-	-	-	-	-	3
stmt	4	-	5	6	-	-	-	-	-	-	-	-
expr	7	7	-	-	-	7	-	-	-	-	-	-
term_tail	9	-	9	9	-	-	9	8	8	-	-	9
term	10	10	-	-	-	10	-	-	-	-	-	-
factor_tail	12	-	12	12	-	-	12	12	12	11	11	12
factor	14	15	-	-	-	13	-	-	-	-	-	-
add_op	-	-	-	-	-	-	-	16	17	-	-	-
mult_op	-	-	-	-	-	-	-	-	-	18	19	-