The Grammar for C-
February 14, 2004

1 Introduction

This is the grammar for C-. This will be the first grammar we use. I have corrected some typos since I first posted it. I have also added the modulo operator "%". I hope to modify the language in a later assignment.

For the grammar that follows Here are the types of the various elements by type font:

- **Keywords are in this type font.**
- **TOKEN CLASSES ARE IN THIS TYPE FONT.**
- *Nonterminals are in this type font.*

The symbol ϵ means the empty string.

1.1 Some Token Definitions

letter = a | ... | z | A | ... | Z

digit = 0 | ... | 9

ID = letter+

NUM = digit+

Also note that **white space** is ignored except that it must separate **ID**'s, **NUM**'s, and keywords. **Comments** are treated like white space. Comments begin with // and run to the end of the line.

2 The Grammar

1. *program* → *declaration-list*
2. *declaration-list* → *declaration-list declaration* | *declaration*
3. *declaration* → *var-declaration* | *fun-declaration*
4. *var-declaration* → *type-specifier* **ID ;** | *type-specifier* **ID [NUM] ;**
5. *type-specifier* → **int** | **void** | **bool**
6. *fun-declaration* → *type-specifier* **ID (** *params* **)** *compound-stmt*
7. *params* → *param-list* | **void**
8. *param-list* → *param-list* **,** *param* | *param*
9. *param* → *type-specifier* **ID** | *type-specifier* **ID [ ]**
10. *compound-stmt* → **{** *local-declarations statement-list* **}**

11. *local-declarations* → *local-declarations var-declaration* | ϵ
12. *statement-list* → *statement-list statement* | ϵ
13. *statement* → *expression-stmt* | *compound-stmt* | *selection-stmt* | *iteration-stmt* | *return-stmt*
14. *expression-stmt* → *expression* **;** | **;**
15. *selection-stmt* → **if** (*expression*) *statement* | **if** (*expression*) *statement* **else** *statement*
16. *iteration-stmt* → **while** (*expression*) *statement*
17. *return-stmt* → **return ;** | **return** *expression* **;**
18. *expression* → *var* = *expression* | *simple-expression*
19. *var* → **ID** | **ID** [*expression*]
20. *simple-expression* → *additive-expression relop additive-expression* | *additive-expression*
21. *relop* → <= | < | > | >= | == | !=
22. *additive-expression* → *additive-expression addop term* | *term*
23. *addop* → + | − | ||
24. *term* → *term mulop unary-expression* | *unary-expression*
25. *mulop* → * | / | % | &&
26. *unary-expression* → *unaryop unary-expression* | *factor*
27. *unaryop* → **!** | −
28. *factor* → (*expression*) | *var* | *call* | *constant*
29. *constant* → **NUM** | **true** | **false**
30. *call* → **ID** (*args*)
31. *args* → *arg-list* | ϵ
32. *arg-list* → *arg-list* **,** *expression* | *expression*