

Program 2: A Lexer for FORTH

CS 636 / 338—Due April 24 2004

Write a primitive lexer for FORTH in C, C++, or Java.

1. Declare an enumerated type for the types of tokens that FORTH has (literal, word, string).
2. Declare a type named Token that consists of a pointer to the string of characters that make up the token, a type code, and the number of times that token occurs in the program.
3. Create an array of type Token to store the tokens that you find. In a real compiler, this data structure would be called the “symbol table”. You may decide to use a separate array for each kind of token
4. Your main program should open the ged source-code file and an output file. Start the output file by writing out your name and assignment number, and the name of the file you are lexing. Then write the heading “Comments”, for the comments that will soon be printed.
5. Read your input file, one line at a time. Skip leading whitespace and find the first token on the line. Proceed as follows:
 - (a) If it is a backslash, write the rest of the line to the output file. Comments do not go into the symbol table.
 - (b) Else if it is a ‘(’, write the ‘(’ and everything that follows it to the output file, ending with the first ‘)’.
 - (c) Else you have a token that belongs in the symbol table. Prepare to allocate dynamic storage for it.
 - i. If the token is a .” find the matching ” and put the whole string into the symbol table, without the .” and without the final ”.
 - ii. Else if the token consists entirely of digits 0..9, mark it as a numeric literal and put it into the symbol table.
 - iii. Else the token is a word. If the word is already in the symbol table, increase the use count. Otherwise create a new symbol-table entry for this token.
 - (d) You are done with this token. If you are at the end of the input line, read another input line. Skip leading whitespace and find the next token. Go back to step (a).
6. When you come to the end of the input file, write a line of dashes to the output file, followed by three lists of tokens (strings, numbers, words). Finally, write out the total number of different tokens in the symbol table.
7. I will supply the test data for this exercise. For my sanity, I want everyone to use the same input file.