#include <iostream>

#include <fstream>

#include <cctype>

#include <cstring> // For memset (needed for string operations like strcpy)

/\* Global declarations \*/

/\* Variables \*/

// 'charClass' stores the current character class: letter, digit, or unknown

int charClass;

// 'lexeme' will hold the current sequence of characters being analyzed

char lexeme[100];

// 'nextChar' will store the current character we're reading from the file

char nextChar;

// 'lexLen' keeps track of how many characters are currently in the lexeme

int lexLen;

// 'token' holds the numeric value associated with the token

int token;

// 'nextToken' will store the next token we are processing

int nextToken;

// 'in\_fp' is the input file stream to read from the file

std::ifstream in\_fp;

/\* Function declarations \*/

// Functions we’ll need to call during lexing

void addChar();

void getChar();

void getNonBlank();

int lex();

/\* Character classes \*/

// These constants help us categorize each character

#define LETTER 0

#define DIGIT 1

#define UNKNOWN 99

/\* Token codes \*/

// These are the different types of tokens we will identify

#define INT\_LIT 10

#define IDENT 11

#define ASSIGN\_OP 20

#define ADD\_OP 21

#define SUB\_OP 22

#define MULT\_OP 23

#define DIV\_OP 24

#define LEFT\_PAREN 25

#define RIGHT\_PAREN 26

/\* main driver \*/

int main() {

// We open the file "front.in" for reading

in\_fp.open("front.in");

if (!in\_fp) {

// If the file cannot be opened, print an error

std::cerr << "ERROR - cannot open front.in\n";

}

else {

// If the file opens successfully, start processing the characters

getChar();

// Keep processing the characters until the end of the file

do {

lex();

} while (nextToken != EOF);

}

return 0;

}

/\* lookup - a function to check for operators and parentheses

and return the correct token \*/

int lookup(char ch) {

switch (ch) {

case '(':

addChar();

nextToken = LEFT\_PAREN; // Token for left parenthesis

break;

case ')':

}