```
    1:
    2: #include <iostream>
    3: #include <fstream>
    4: #include <iomanip>
    5: #include "SyntaxAnalyzer.h"
    6:
    7: int main()
    8: {
    9:     const int COL_SIZE = 20;
   10:
   11:     std::ifstream fin;                    // Input file stream
   12:     std::string inFile;                   // Input file name
   13:     std::vector<Lexer::Token> lineTokens; // List of lineTokens
   14:     std::vector<Lexer::Token> tokens;
   15:     std::stringstream *buffer;
   16:     std::string line;
   17:
   18:     std::vector<std::string> files = {"test1.txt", "test2.txt", "test3.txt"};
   19:
   20:         for (std::string file : files)
   21:         {
   22:
   23:                 // Open the file
   24:                 fin.open(file.c_str());
   25:
   26:                 if (!fin)
   27:                 {
   28:                         std::cout << "file not found" << std::endl;
   29:                         continue;
   30:                 }
   31:
   32:                 std::cout << std::endl
   33:                         << "RUNNING TEST CASE FILE \"" << file << "\"" << std::end
l
   34:                         << std::endl;
   35:
   36:                 // File has opened, instantiate the lexer.
   37:                 Lexer *lexer = new Lexer();
   38:
   39:                 int lineNumber = 1;
   40:
   41:                 while (getline(fin, line))
   42:                 {
   43:                         buffer = new std::stringstream(line);
   44:                         lineTokens = lexer->lex(*buffer, lineNumber);
   45:
   46:                         tokens.insert(tokens.end(), lineTokens.begin(), lineTokens
.end());
   47:
   48:                         lineNumber++;
   49:                 }
   50:                 fin.close();
   51:
   52:                 std::ofstream out;
   53:                 out.open("output.txt");
   54:
   55:                 SyntaxAnalyzer *syntaxAnalyzer = new SyntaxAnalyzer(tokens, out, t
rue);
   56:
   57:                 try
   58:                 {
   59:                         // Run syntactical analysis
   60:                         syntaxAnalyzer->Analyze();
   61:                 }
   62:                 catch (const SyntaxError &e)
   63:                 {
   64:                         out << std::endl << "ERROR: " << e.getMessage();
   65:                 }
   66:
   67:                 tokens.clear();
```

```
68:         }
69:     std::cout << std::endl
70:               << "EXECUTION HAS COMPLETED." << std::endl;
71:     std::cout << "Press enter to continue. . ." << std::endl;
72:     std::cin.get();
73:
74:     return 0;
75: }
```