NAME: Ben Puryear
FILE: hw3-writeup.pdf
DATE: Spring 2023
DESC: This pdf will provide a short description of any challenges and/or issues I faced and how I addressed them in this assignment.

Challenges faced:

I faced many challenges with my implantation of the Lexer from HW2. There were many edge cases that I did not test for that I had to go back and fix. The main one involved parenthesis that came after an ID with no space in between them. Solving this problem instantly made the program pass an extra 5 tests.

I also faced a challenge whenever the Parser needed to look ahead 2 tokens to decide what function to call. To solve this, I would group those two functions together, making sure it MUST be one of them. After that I would eat the common Token, then use match to determine which function to call.

Testing:

All Tests Pass

```
[       OK ] BasicSimpleParserTests.TestAdded2 (0 ms)
[ RUN      ] BasicSimpleParserTests.TestAdded3
[       OK ] BasicSimpleParserTests.TestAdded3 (0 ms)
[ RUN      ] BasicSimpleParserTests.TestAdded4
[       OK ] BasicSimpleParserTests.TestAdded4 (0 ms)
[ RUN      ] BasicSimpleParserTests.TestAdded5
[       OK ] BasicSimpleParserTests.TestAdded5 (0 ms)
[----------] 75 tests from BasicSimpleParserTests (3 ms total)

[----------] Global test environment tear-down
[==========] 75 tests from 1 test suite ran. (3 ms total)
[  PASSED  ] 75 tests.
(base) benpuryear@Bens-MacBook-Pro-2 hw3-Ben10164 %
```

Valgrind Output

```
[  PASSED  ] 75 tests.
==1823436==
==1823436== HEAP SUMMARY:
==1823436==     in use at exit: 0 bytes in 0 blocks
==1823436==   total heap usage: 2,865 allocs, 2,865 frees, 271,327 bytes allocated
==1823436==
==1823436== All heap blocks were freed -- no leaks are possible
==1823436==
==1823436== For lists of detected and suppressed errors, rerun with: -s
==1823436== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
bpuryear@ada:~/hw3-Ben10164$
```

Explanation of Added Tests

TestAdded1: Checks to make sure that if a function does not have a closing parenthesis, it fails.
TestAdded2: Checks to make sure an integer declaration must have a ID following it.
TestAdded3: Checks to make sure a while loop can't rely on an INT_TYPE as its statement.
TestAdded4: Checks to make sure not having a complete expr in the for loop causes an error.
TestAdded5: Checks to make sure an out of place brace throws an error.

Additional Tests

```
TEST(BasicSimpleParserTests, TestAdded1) {
  stringstream in("void f(");
  try {
    SimpleParser(Lexer(in)).parse();
    FAIL();
  }
  catch(MyPLException& e) {
    string msg = e.what();
    ASSERT_EQ("Parser Error: ", msg.substr(0, 14));
  }
}

TEST(BasicSimpleParserTests, TestAdded2) {
  stringstream in("void f() { while (i) { int } }");
  try {
    SimpleParser(Lexer(in)).parse();
    FAIL();
  }
  catch(MyPLException& e) {
    string msg = e.what();
    ASSERT_EQ("Parser Error: ", msg.substr(0, 14));
  }
}

TEST(BasicSimpleParserTests, TestAdded3) {
  stringstream in("void f() { while (int) { return false } }");
```

```
  try {
    SimpleParser(Lexer(in)).parse();
    FAIL();
  }
  catch(MyPLException& e) {
    string msg = e.what();
    ASSERT_EQ("Parser Error: ", msg.substr(0, 14));
  }
}

TEST(BasicSimpleParserTests, TestAdded4) {
  stringstream in("void f() { for(int i = 0; i < ; i = i + 1) { } }");
  try {
    SimpleParser(Lexer(in)).parse();
    FAIL();
  }
  catch(MyPLException& e) {
    string msg = e.what();
    ASSERT_EQ("Parser Error: ", msg.substr(0, 14));
  }
}

TEST(BasicSimpleParserTests, TestAdded5) {
  stringstream in("void f()}");
  try {
    SimpleParser(Lexer(in)).parse();
    FAIL();
  }
  catch(MyPLException& e) {
    string msg = e.what();
    ASSERT_EQ("Parser Error: ", msg.substr(0, 14));
  }
}
```