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.

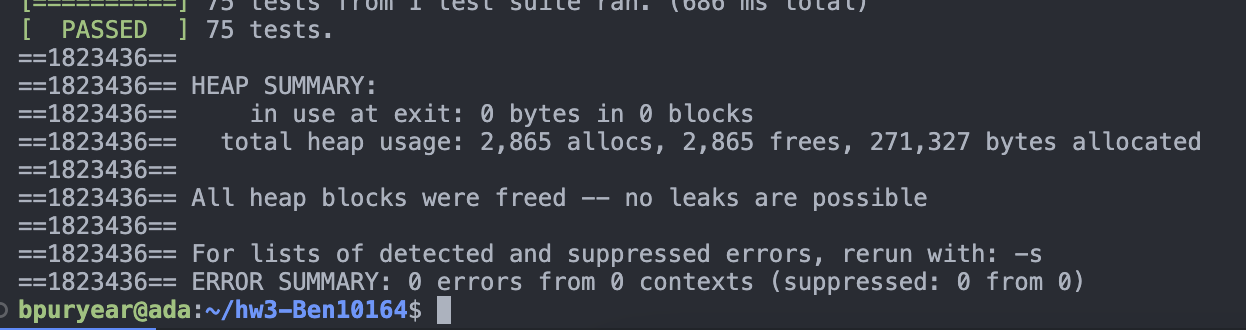
**Testing:**

All Tests Pass

Text

Description automatically generated

Valgrind Output



**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));

}

}