Maxwell Schaefer
Email: mrs314@case.edu
Course: CSDS 337 - Compiler Design
Instructor: Dr. Vipin Chaudhary

Homework - #
ID: mrs314
Term: Spring 2023
Due Date: 12^{th} April, 2023

Number of hours delay for this Programming Assignment: 22
Cumulative number of hours delay so far: 36

I discussed this homework with:

Assignment

For this assignment, you're given partially completed code of a compiler which uses an abstract syntax tree; your job is to complete it. By the end, you will have a fully functional compiler. You will need to fill in certain parts of the AST code and parts of the parser which generates the AST. Both of these should be somewhat familiar: the parser is very similar to the one you wrote in PG-2 (the only difference is that code actions now generate an AST). The AST itself uses LLVM for code generation, which should be familiar from PG-3.

Problem 1

Fill in code actions for the following nonterminals in `src/frontent/parser.y`:

- `type` (5 pts)
- `funDef` (15 pts)
- `selStmt` (5 pts)
- `iterStmt` (for this one, you also should modify the grammar to support for loops) (10 pts)
- `jumpStmt` (5 pts)
- `expr` (5 pts)
- `andExpr` (5 pts)

All of the other code actions have been done for you, so you can use them as examples. When you are finished with this, all grammar productions in the parser should have a code action. See the comments in `parser.y` for more information, guidelines, and hints.

Problem 2

Fill in the following parts of the AST (note that you may need to finish all for compiling to work). By running the `./executeTests.sh` script, you can compile and run all the samples in the `tests` folder. You should add your own to test your compiler changes. You also may need to declare header files for their corresponding C++ files if they do not exist:

- `expressions/sub.cpp` (3 pts)
- `expressions/bool2Int.h` (2 pts)
- `expressions/comparison.cpp` (3 pts)
- `expressions/int2Bool.cpp` (3 pts)
- `expressions/multiplication.cpp` (3 pts)
- `expressions/division.cpp` (3 pts)

- `expressions/negative.cpp` (3 pts)
- `expressions/bool.cpp` (5 pts)
- `expressions/and.cpp` (10 pts, make sure to not evaluate 2nd arg if 1st is false. Also, NO STACK ALLOCATIONS ARE ALLOWED HERE!)
- `statements/for.cpp` (15 pts)

Documentation on the structure of the AST and more info about each part is available in the README file.
