

CMSC 430 Programming Exercise

TypeChecker/AST

Due Date: Friday Nov 1st, 11:59pm.

This project is the second in the series of projects that you have to implement in order to build a compiler for 'C-', a variant of 'C'. In this project you will finish implementing the compiler front end for C-, adding symbol tables, type checking, and creating an abstract syntax tree (AST) representation of the input.

Getting Started

To get the materials for this project, type the following line to copy over the files.

```
cp -r ~ctseng/proj3 ~/proj3
```

Look at the file README for more detailed directions and hints.

Requirements

Symbol Tables C- applies nested lexical scoping, where every pair of curly brackets creates a new nested scope (and can include new variable declarations). Variables visible include those declared locally and in enclosing scopes. You must create and maintain a set of nested symbol tables to support nested lexical scoping found in C- programs. Every variable and function declaration must be entered in the appropriate symbol table so the information may be used in the type checker (for this project) and code generator (for the next project). You should use the symTab and symTabEntry classes provided.

Type Checker The C- type checker ensures all variables accessed are legally declared and have proper types. Warn of undefined variables or variables defined multiple times. A variable can be local or global or passed as a parameter. The type checker should perform the following operations: check operand types, ensure boolean operands where required, find undefined variables, find multiply defined variables, check function parameters/arguments, check forward declarations match function declarations, check return types match function declarations.

AST Generation The final task of the C- front end is to generate an intermediate representation of the program, in the form of an AST. The AST will be used in future projects for generating Java byte codes and performing code optimizations. You should use the expNode and astNode classes provided.

Submission Instructions

You can turn in your assignment using the `submit` program described in previous projects. To submit, go to your directory containing the code for project 3 and type:

```
submit 3 mycc.lex mycc.cup *.java
```

The submit program will accept multiple submissions up to the submission deadline, overwriting previous submissions. Feel free to submit your project as many time as you desire before the deadline.

The late submission policy is: 20% penalty for first 24 hours, 10% each additional day. Maximum 5 days late.