Stellenbosch UNIVERSITY IYUNIVESITHI UNIVERSITEIT

RW745 Software Language Engineering

Assignment 3 Due on Apr 15, 2024, 17:00

Stellenbosch University Computer Science Assignment Declaration

I know that using another person's ideas (including written work and source code) and claiming them as my own constitutes plagiarism. I am aware that the potential penalties for this offence, as specified in the University's policy on academic integrity, may include expulsion from the University and other serious consequences.

1. This submission is entirely my own work, excepting the inclusion of resources explicitly permitted for this assignment, and assistance as noted in the following item. I have not used any generative AI technology or tools (such as ChatGPT), unless explicitly permitted in the assignment.

2. My submission acknowledges the source of all libraries and external sources used, and identifies any other students and/or staff (including demis, tutors, and lecturers) with whom I have discussed this assignment, as well as the extent of those discussions. Note that, unless explicitly stated otherwise, submissions or solutions to assignments in previous courses at this University or any other educational institution which even partly correspond to this assignment may not be used as external sources; in particular, do not look at or use solutions to assignments from previous years.

3. I have not allowed, and will not in the future allow, anyone to copy any portion of my work, or give them access to it in any way. In addition, I will not make my work publicly available in any way, including posting my code in public source code repositories or forums, unless explicitly permitted by the lecturer.

4. I have not and will not facilitate plagiarism, such as by distributing any written work or source code created by a fellow student.

5. I understand that any code I submit may be inspected for plagiarism detection (either manually or by automated systems) and be retained for detecting plagiarism in other courses.

US number / *US nommer* _____

Signature / *Handtekening* _____

Please fill out and submit a scan or photo together with your homework. Submissions without signed declaration will not be marked.

Question 1.
Consider the language Alan used in the first assignment, and its specification in the document on the sunlearn pages. In this assignment you should extend the language and implement a checker for the extended scope and type rules, as specfied below.

1. Extend your grammar and parser developed in the first assignment to allow
   - top-level variable declarations (i.e., global variables) that precede the top-level body; and
   - nested function declarations (i.e., functions within functions).

   Note that variable and function declarations can be mixed arbitrarily, but declarations cannot be mixed with statements. Your language extension should follow the current style of the Alan syntax and not introduce new keywords. **10 marks.**

2. Implement a symbol table and scope rule checker for the extended Alan version. The scope rules are given by the following principles.

   scopes: the program and each function declaration constitute a new scope.
   scope nesting: scopes are nested into each other according to the syntactic structure of the program.
   shadowing: declarations in inner scopes shadow (i.e., overwrite) declarations for the same name in outer scopes.
   transparency: declarations in outer scopes are visible in any enclosed inner scopes, unless they are shadowed.
   no persistence: declarations are invisible outside the scope in which they are immediately contained.
   declare before use: identfiers must be declared by a visible (i.e., textually preceding in the current or any enclosing scope but not shadowed) declaration before they can be used.
   no redeclaration: each scope can contain at most one declaration for each identifier.
   single namespace: within a given scope, each visible identifer can only refer to the program, a function, or (scalar or array) variable.

   Your scope rule checker should identify all scope rule violations in one pass; in particular, multiple uses of an undeclared identifier should produce multiple error messages. It should not check for type errors, e.g., a multiplication of an identifier that refers in the current scope to a function and one that refers to an array variable should not produce an error message. **50 marks.**

3. Implement the type checking rules, as specified in the original Alan language specification. Extend the given type checking rules in a consistent fashion, if necessary. Your type rule checker should identify all type rule violations in one pass. It should not produce type error messages for undeclared identifiers, even if they are used inconsistently in a given scope, e.g., as a function and an arrray variable. **40 marks.**

Notes:
1. Contact me if you did not complete the first assignment, or if you prefer not to work from your own submission. You will be given the memo solution, but incur a 10 marks penalty.
2. Note that your marks for Part 2 will be capped at 20 marks if you do not handle global variables and nested function declarations.
3. Note that Part 3 requires at least a rudimentary working solution of Part 2. However, errors in Part 3 that are caused by errors in Part 2 will not be penalized again.

4. You must implement two separate listener or visitor classes for Part 2 and Part 3.
5. You must implement a class `alan` whose main method parses the program and calls the scope and type rule checker. Your implementation should be able to handle programs with syntax errors.
6. Your implementation must write error messages to standard error channel. The error messages must give precise location information, at least the location of the first character of smallest sub-tree enclosing the error. The error messages from the scope and type rule checkers must appear in order of the respective source locations.

Submission:

1. Submission deadline is **Apr 15, 2024, 17:00**.
2. Please submit all *source* files that are required to build a running version; please don't submit any ANTLR jars.
3. We will be using ANTLR v4.13.1 and Java 17 for marking.
4. Please submit all files on your Gitlab repository, even if you are re-using files from prior submissions without any changes. Please create a directory `assignment3` for this submission.
5. Please submit a pdf of your signed declaration statement.
