COS341 Practical 3: Academic Year 2023

This year's version of the students' programming language SPL shall have the following **scoping rules**:

- **For variables there shall be only one global scope** – i.e.: anywhere in an SPL program, any two variables with the same name refer to the same computational entity – or, in other words: every variable is a global variable.
- **Different scopes however are applicable for procedure names** as follows:
  - The Main program may call only those procedures (identified by their names) which are declared as immediate "child" procedures directly "underneath" Main; it may not call "deeper" to procedures declared as children of the children, etc.
  - Ditto also for procedures "under" Main: they may only call their children but not their grand-children.
  - Sibling procedures at the same level of the hierarchy may call each other (which allows for mutual recursion) as well as also themselves (which allows for self-recursion).
  - "Upward" calls (from child procedure to parent or grandparent procedure are not allowed (i.e.: there is also no "vertical" mutual recursion in which parent and child procedures recursively call each other).
  - Also not allowed are any calls to "uncle", "nephew", "cousin" procedures in the hierarchical tree of procedure declarations.
  - *For illustration see the figure attached.*

These rules give rise to the following **Naming conventions** for SPL procedures:

- Procedures which can call each other (according to the semantic rules of above) cannot have the same names.
- Procedures which cannot call each other (according to the semantic rules of above) can have the same names (though they are internally treated as different computational entities): they "live" in different scopes.
- No procedure can have a child and a sibling with same Name.

Accordingly, **two different kinds of semantic errors** could possibly arise in some SPL program:

- ERROR: *"The procedure called here has no corresponding declaration in this scope!"*
- WARNING: *"The procedure declared here is not called from anywhere within the scope to which it belongs!"* (In other words: an un-used procedure is merely "dead code".)

Your Tasks:

- Make a Hash-Table as a symbol table in such a manner that the table's Hash Keys correspond to the Unique Node IDs of a parsed SPL program's AST (abstract syntax tree as per Practical 2).
- Given an AST (from parsing), analyse it (by crawling up and down its branches) to identify the procedure scope-regions which which each AST-node belongs.
- Provide unique Scope ID numbers accordingly: different Scopes have different Scope ID numbers.
- For each AST node, store its corresponding Scope ID number in the symbol table (i.e.: the Hash Table).
- Visualize the Symbol Table as an HTML table, such that the Tutors can see the contents of the table by means of a web-browser.
- IF the AST (from parsing) contains any of the two above-mentioned semantic problems (call to an un-declared procedure, or declaration of an un-used dead-code procedure), then the corresponding Error/Warning messages must also be displayed.

Turn the page →

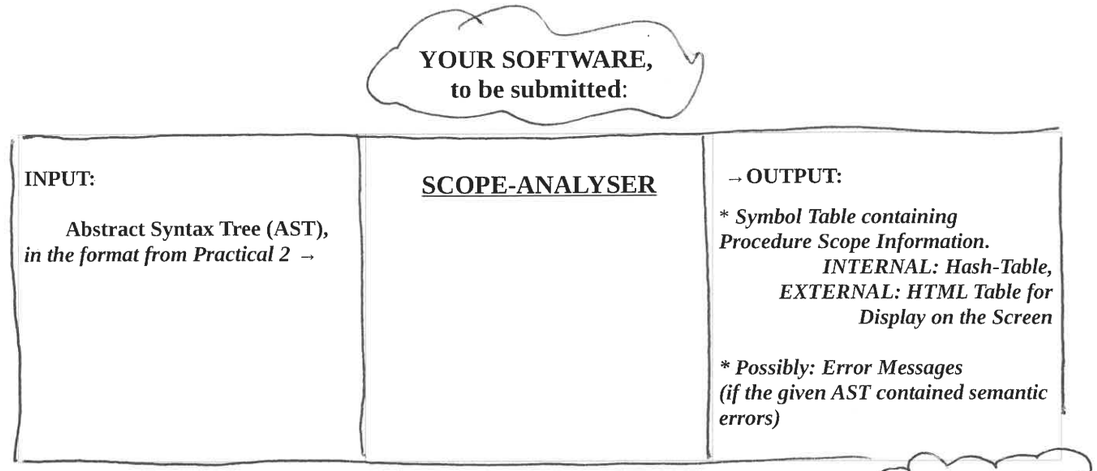
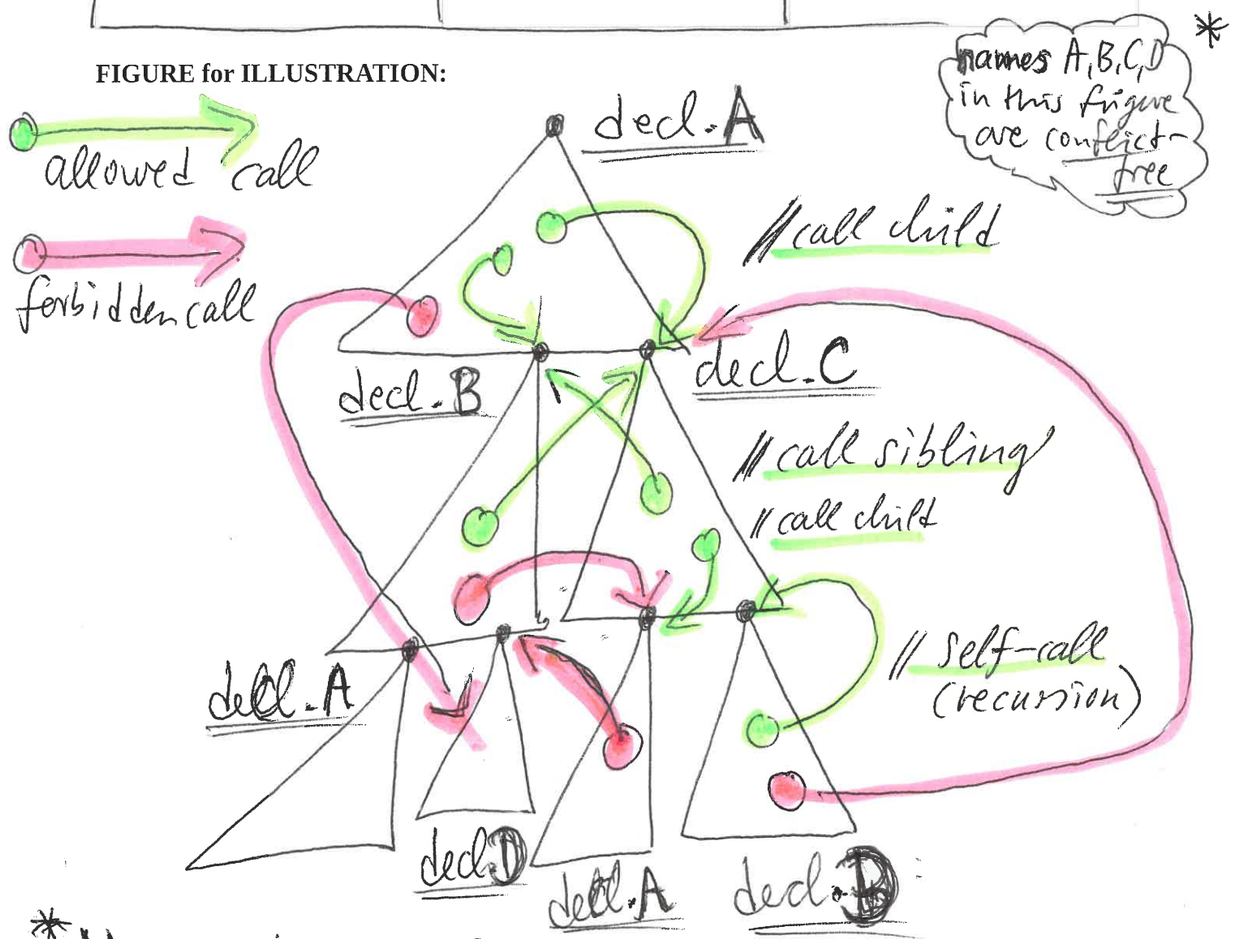


FIGURE for ILLUSTRATION:



And now: **HAPPY CODING :)**