Final Assessment

In this assessment we will be testing your ability on understanding and implementing the lexical, syntax, and semantic analysis portion of the compilation process. This is done with the creation of a lexical analyzer, either top-down or bottom-up parsers, a symbol table, and a method for analyzing static semantics. This will allow you to choose between practical and conceptual problems.

PRACTICAL

(50 Points)

1) In this problem you will be designing code that should be able to take in a file that is analyzed for lexical and syntactical correctness. This programming language should allow for the following type of statements:

- Statically typed identifiers (variable/function/method) definitions
  - functions with return types
  - Variables that are declared with their data types that cant be changed
- Class definitions
- Void Datatype
- Int Datatype
- String Datatype
- Floating Point Datatype
- Boolean Datatype
- Assignment Statements
- Expressions
- Boolean Expressions
- Class dereferencing of variables and methods

(25 Points)

2) In this problem you will be designing code that should be able to run along side of the lexical, syntax, and semantic analyzer; this code will track variable names, types and scoping (assuming static scoping). This portion is the creation of a symbol table, which are used in semantic analysis primarily for type checking and variable scoping.

(25 Points)

3) Create and program an attribute grammar with predicate rules, and logic that doesn't allow for any mismatch assignments of expression types to incorrect variable types, doesn't allow the dividing of zero, coerces all Booleans to integers for mathematical operations, converts all integers to floats for binary operations with type mistaches.

Conceptual – Choose remaining 50 points

1. 25 points
   Create the formal definition of the attribute grammar discussed in Practical Problem 3.
   - 10 points
   Create a fully decorated tree for the following lines of code
      a. int x = 5.0 + “string”;
      b. int y = 0, x = y + 2 * (y * true);
      c. string y = (“Hello World”).substr(5);
   - 15 points

2. 25 Points
   Prove Total correctness of the following code block, and list all axioms and inference rules used to determine this:

```
double answer = 1, d = 0.00001;
while (answer*answer <= num){
    answer += d;
}
if (answer*answer > num) {
    answer -= d;
}
// { answer = sqrt(num) }
```

3. (10 points)
   Convert the following context-free grammar to attributes grammar in order to recognize strings that consist of n number of “a” followed by 3n number of “b” followed by 2n number of “c,” where n >= 1. The strings “abbbcc” and “aabbbbbbcccc” belong to this grammar, but the strings “aaabbbbcc” and “aabbbcc” do not.

   <letter-sequence> ::= <a-sequence> <b-sequence> <c-sequence>
   <a-sequence> ::= a | <a-sequence> a
   <b-sequence> ::= b | <b-sequence> b
   <c-sequence> ::= c | <c-sequence> c

4. (10 points)
   Find the weakest P and show your work:

   - { P } if (x > y) y = 2 x + 1 else y = 3 x – 1 { y > 3}
   - { P } x = y + 10 { y > 10 }
   - { P } y = y + 1; z = x + y; { x < z }
   - { P } x = 3 * y + 1; y = x – 4; { y < 0 }
   - { P } If (x > 0) then y = y + 2 else y = y + 3 { y ≥ 0 }
   -

5. 10 points
   Give an operational semantic definition of the following:
   - Java do-while
   - Ada for
   - C++ if-then-else
   - C for
   - C switch

6. 20 points
   Give an denotational semantic definition of the following:
   - Python for
   - Java class
   - C++ function
   - C switch
   - Java Boolean expressions