Programming Languages

Qualifying Examination
Fall 2014
End of Semester Session

November 14, 2014

Syntax and Semantics [17 Pts]

Answer the following questions:

1. Provide an example of a syntactic structure of a programming language that is LR(1) but not LL(1); justify your answer. (Hint: the feature is common among almost all programming languages.) If you cannot think of such feature in programming languages, instead you can provide a formal language (with justification) that is LR(1) but not LL(1).

2. Provide a regular expression that describes floating point numbers (account for sign and use scientific notation).

Data Types [17 Pts]

Answer the following questions:

1. Describe the concept of dangling pointer; provide a simple code fragment in C++ that generates an instance of a dangling pointer.

2. Describe the tombstones method to address the dangling pointer problem.

3. Describe the mark-sweep and the reference-counters methods; compare them in terms of relative strengths and weaknesses.

Subprograms [17 Pts]

Programming languages like Pascal and Modula-2 allow nested definitions of procedures. Answer the following questions:

1. Create an example that uses nested procedures to demonstrate the different results that can be obtained by using static scoping vs. dynamic scoping;
2. Describe precisely the data structures that are used in order to implement nested procedures with static scoping;
3. Provide an example that illustrates the difference between deep and shallow access in the implementation of dynamic scoping.

Object Oriented Programming [17 Pts]

Answer the following questions:

1. Define the concept of multiple inheritance; contrast it with single inheritance and identify at least one language which supports multiple inheritance;
2. Describe the "diamond" problem of multiple inheritance;
3. Java does not directly support multiple inheritance, but it allows one to simulate it (what Sebesta calls *mix-in inheritance* in his book); describe such method, provide an example, and identify the difference with respect to multiple inheritance.

Functional Programming [17 Pts]

Consider the distinction between **strict** and **non-strict** functional languages:

1. Define the two concepts;
2. Illustrate with an example a situation where the two can produce distinct results;
3. Discuss one situation where non-strict languages allow the encoding of problems that cannot be effectively dealt with using a strict language (provide a meaningful example).

Language Design and Implementation [15 Pts]

We would like to invent a programming language which allows the use of nested procedures (with static scoping) along with unconditional goto instruction. The use of the goto statement is completely unrestricted (i.e., you can jump anywhere in the program).

1. Identify what kind of problems can arise from the interaction of static scoping and goto.

2. Let us now consider a flat language (like C—where we do not allow nested procedures); design an implementation solution of unrestricted goto statements that preserves (in the best possible way) correctness of the execution.