COL728 Minor2 Exam
Compiler Design
Sem II, 2017-18

Answer all 5 questions Max. Marks: 20

1. Short questions

a. Give an example of a program that is not a legal program if we assume static scoping, but is a legal program if we assume dynamic scoping. i.e., the same program should be well-scoped if we assume dynamic scoping but not well-scoped if we assume static scoping. Use a C-like language syntax [2.5].

1b. Give an example to show that “stronger type system may force the programmer to write a slower program implementation for the same program logic”. In your example, you need to show two languages, one with a weaker type system than the other. Show that the same program logic may be slower to implement in the second language than in the first language. By “slower to implement”, we mean that the runtime of the implemented program is slower. The relative slowness of the implementation should be directly related to the stronger type system. In other words, the stronger type system should force the programmer to write a slower implementation.

Ideally, you should select from well-known languages (to make the comparison). If you decide to describe your own language (with its type system), the description needs to be very clear. [2]

1c. Consider the following two versions of the typing rule for the assignment statement in a C-like imperative language:

O |- x:T1
O |- e1:T1
---------- [Assignment-rule-1]
O |- x=e1 : T1

O|-x:T0
O|-e1:T1
T1 <=T0
---------- [Assignment-rule-2]
  |- x=e1 : T1

Give an example of a program that would be type-incorrect if one rule is used, but would be type-correct if the other rule is used. Mention which of the rules would cause that example program to be type-incorrect, and which of the rules would cause the same example program to be type-correct. [2.5]

1d. Explain how cascading errors can be avoided during type checking by defining a dummy type called “No_type”, such that No_type <= T for all types T. Use an example to demonstrate this. [3]

2. Consider the following programming language grammar (same as that discussed in class):

P --> D; P | D

D --> def id(ARGS) = E
ARGS --> id, ARGS | id
E --> int | id | if E1 = E2 then E3 else E4 | E1 + E2 | E1 - E2 | id(E1,...,En)

Let NT(e) be the number of temporaries needed to evaluate e (as discussed in class).

a. How does performing this computation of NT(e) allows faster code generation on a stack machine? [1]

2b. Write the code-generation procedure for function dispatch (caller and callee).

cgen(def f(x1,x2,...,xn) = e) = ?

cgen(f(e1,e2,...,en)) = ?

You can use the same (or different) code generation strategy as discussed in class, but something that takes into account the calculation of NT(e). In either case, clearly specify the code-generation strategy: e.g., what are the invariants, how you implement them, etc. [4]

3. Multiple Inheritance : Consider the following class hierarchy. What would be a reasonable code generation strategy (something that is simple and performant) for a hierarchy like this. Hint: clearly specify if you are assuming virtual or replicated inheritance. [3.5]

Class A { …. }
Class B inherits A { … }
Class C inherits A { … }
Class D inherits B inherits C { …. }

4. List at least two applications of formally-specified operational semantics for a programming language. Be as clear and precise as possible. [1.5]