CSCE 531 Fall 2001

MIDTERM EXAM–PARTIAL CORRECTION GUIDE
Friday 00/9/28—Two Pages, Closed Book

1 Short Questions–1 point each; 12 points total

1. What is the difference between a translator and a compiler? **Answer:** A compiler is a kind of translator whose source language is high level and whose target language is machine code (or assembly code).
2. What is the name for the source language of the compiler that we are building in CSCE 531? **Answer:** Triangle
3. What is the name of the machine ("target machine") whose language is the target language of the compiler that we are building in CSCE 531? **Answer:** Triangle Abstract Machine (TAM)
4. What is the difference between the von Neumann architecture and the Harvard architecture? **Answer:** The von Neumann architecture has a single memory (store) for code and data; the Harvard architecture has separate code and data memories
5. The TAM has a Harvard architecture. True or false? **Answer:** True
6. The TAM is a stack-based machine. True or false? **Answer:** True
7. The TAM has no data registers. True or false? **Answer:** True
8. Why is it easier to write a compiler for a target machine with no registers? **Answer:** Because there is no register allocation problem to solve.
9. What are the three components of the state in the denotational semantics approach[1]? **Answer:** mem, i, o
10. What is a cross-compiler? **Answer:** A cross-compiler is a compiler that runs on a machine different from its target machine.
11. BNF production rules may describe contextual languages. True or false? **Answer:** False: only context-free languages.
12. BNF production rules and regular expressions have the same expressive power. True or false? **Answer:** False: BNF correspond to context free languages. Regular expressions correspond to regular languages. All regular languages are context free, but some context-free languages are not regular.

[1] Every question on denotational semantics refers to the simple language described in class.

2 Semantics–10 points

1. (3 points) In denotational semantics, the memory function, *mem* is a function from the set of *mysteries1* to the set of *mysteries2*. What are *mysteries1* and *mysteries2*? **Answer:** identifiers, integers (also acceptable: identifiers, values).

2. (2 points) *Aliasing* is the situation in which two identifiers refer to the same memory location at the same time. Why is aliasing difficult to capture in denotational semantics? **Answer:** Because one has to enforce the requirement that two identifiers have identical values at all times.

3. (5 points) Describe (very briefly) the semantic difference between commands, expressions, and declarations. **Answer:** A command is executed to update variables or to perform I/O. An expression is evaluated to yield a value. A declaration is elaborated to produce bindings [textbook, pp.18-19].

3 Tombstone Diagrams–10 points

Suppose you have a compiler from CLU to x86 written in CLU and a compiler from CLU to x86 written in x86. You need to write a compiler from CLU to Utopia-1.

1. (3 points) Draw tombstone diagrams describing the two compilers you have.

2. (2 points)You write a compiler from CLU to Utopia-1 written in CLU. Draw a tombstone diagram describing this compiler.

3. (5 points) Show how you would obtain the desired compiler. Use tombstone diagrams to illustrate the steps you follow. Identify the bootstrap step. Be concise.

Answer: This is precisely the half-bootstrap pattern of Example 2.19 in the textbook (replacing Ada with CLU).

4 BNF–6 points

1. (2 points) In Pascal, an identifier is a non-empty sequence of letters and numbers that starts with a letter. Provide BNF production(s) to describe Pascal identifiers. **Answer:**

```
<id> ::= <letter> | <letter> <letters-or-digits>
<letters-or-digits> ::= <letter> | <digit> | <letter> <letters-or-digits>
                      | <digit> <letters-or-digits>
```

2. (2 points) Provide a regular expression that describes Pascal identifiers. **Answer:** `[a-z] ([a-z]|[0-9])*`

3. (2 points) Provide a deterministic finite state automaton that recognizes Pascal identifiers.

5 Compilation–12 points

1. (2 points) List the three phases of compilation. **Answer:** Syntactic analysis, static semantics analysis (a.k.a. contextual analysis a.k.a. contextual constraints), code generation.

2. (2 points) What are the two kinds of constraints that are checked in contextual analysis? **Answer:** scope and type constraints.

3. (4 points) Briefly contrast static and dynamic scope rules on the following program:

```
int x = 5; // a global variable
main
 {
 foo  // a parameterless function
  {
  int x = 4;
  ...
  }
 bar  // another parameterless function
  {
  call foo;
  write x;  // which x?
  ...
  }
```

   Answer: 5 is printed under static rules; 4 under dynamic rules.

4. (2 points) Why is it impossible to write a one-pass compiler for Java? **Answer:** Because variables may be declared after they are used.

5. (2 points) The argument you gave in answering the previous question does not hold for Pascal. Why? **Answer:** Because identifiers must be bound before they are used.