College of Computer Science
Northeastern University
COM 1721: Freshman Honors Seminar

Fall 2002
Handout 17
17 December 2002

Sample Solution to Quiz 5

1. Parsing

Recall the arithmetic expression language that we covered in class and in a programming assignment. The production rules and evaluation rules were given as follows:

$$\begin{array}{ll} R0: \langle e \rangle \rightarrow \text{num} & V(\langle e \rangle) = \text{num} \\ R1: \langle e \rangle \rightarrow \text{op}\, \langle e \rangle_1\, \langle e \rangle_2 & V(\langle e \rangle) = \text{op}(V(\langle e \rangle_1), V(\langle e \rangle_2)), \end{array}$$

where num represents a number, op represents an arithmetic binary operation and $V()$ represents the value function.

(a) Draw the parse tree for the following valid expression and determine the value.

$$* - * 5\ 2\ /\ 12\ 3 + 6\ 3$$

Answer: The order of operations is shown here. The corresponding parse tree can be constructed in a straightforward manner.

$$[* \{- (* 5\ 2)\ (/\ 12\ 3)\}\ (+\ 6\ 3)]$$

(b) The above grammar is *unambiguous* in the sense that the parse tree (and, hence, the value) of a given expression is unique. Suppose we add the following third production rule to the language, that allows the unary '$-$' operation.

$$R2: \langle e \rangle \rightarrow -\langle e \rangle_1 \quad V(\langle e \rangle) = -V(\langle e \rangle_1)$$

The enhanced grammar now becomes ambiguous; that is, there may be more than one way to parse a given expression. Describe two different ways to parse the following expression (either using parse trees or giving the order of operations), and show that the two different parsings yield two different values.

$$* - - 7\ 10\ 5$$

Answer: Two different legal ways to parse the expression are:

$[* \{- (-\ 7)\ 10\}\ 5] = -85.$

$[* \{- (-\ 7\ 10)\}\ 5] = 15.$

2. Program memory management

What is the *buffer overrun* problem? What is the difference between a reference counting garbage collector and a tracing garbage collector?

Answer: The buffer overrun problem occurs when data, written for an object, overflows the memory allocated for the object and possibly overruns into memory allocated for some other object, thus resulting in an incorrect (and possibly dangerous) execution of the given program.

Reference counting keeps a count of the number of references made to each object. Whenever a reference to an object is added, the count for the object is increased. Whenever a reference to an object goes out of scope or is reassigned, the count is decremented. When an object is garbage collected, the counts of all objects it references are decremented.

Tracing collectors, on the other hand, traverse the pointers from the root set (global variables, stack, etc.) at the time of garbage collection. All reachable objects form the "live space" and are copied over to a contiguous region of the heap. Unreachable space can be reclaimed. A tracing collector does not suffer from a drawback of the reference counting collector of failing to reclaim space used by objects that have non-zero reference counts and, yet, are not reachable from the root set.