ECE351 Sample Questions (Second Set): CFGs, Parsers, Semantic Analysis, Memory layout, Optimization

Question 1: True/False questions (20 points)

Please specify if the following assertions are True or False. Each sub-question is worth 2 points.

1. Is the following grammar is ambiguous

$$E \to EaE$$
$$E \to b$$

   Answer: True. (draw different parse trees for the string babab.)

2. Precendence rules help overcome ambiguity in context-free grammars

   Answer: True.

3. For a bottom-up parser G, a string s is said to be accepted by G if there exists a leftmost derivation in reverse starting from the string back to the start symbol of G.

   Answer: False. (Bottom-up parsers perform a rightmost derivation in reverse.)

4. Lookahead is an important step in LL(k) parsers. In such parsers, the parsing algorithm can look ahead as many tokens as needed to correctly parse an input string.

   Answer: False. (LL(k) lookahead atmost 'k' symbols.)

5. Shift-reduce parsers only use two actions, namely, shift and reduce. They shift tokens into a parsing stack, and 'reduce' when any sub-string (called a handle) on the parsing stack can be reduced using a production rule.

   Answer: False. (It reduces only if the appropriate sub-string on the stack (called a handle) is found on top of the stack, and not inside.)

6. Global variables are typically stored on the call-stack.

   Answer: False.

7. All buffer overflow attacks can easily be protected against by making the call-stack non-executable.

   Answer: False.

8. The register allocation problem has a known (as of March 2014) polynomial time algorithm.

   Answer: False.

9. In modern garbage collection algorithms, it is possible that some reachable memory (reachable from a root set of locations at runtime) can also be garbage simultaneously.

   Answer: True.

10. The global copy propagation algorithm studied in class takes time exponential in the number of basic blocks of the control-flow graph of the program-under-analysis.

    Answer: False.

Question 2: Short answer questions (30 points)

For each of the questions below, provide a short, correct and complete answer. Each question carries a maximum of 5 points.

1. The following C/C++ code has an integer overflow. Does it necessarily lead to a buffer overflow? If yes, explain how. If no, then subtly modify the code appropriately to cause a buffer overflow. You are only allowed to modify (add or delete) a single character in the program. You are also not allowed to change the function declaration in any way. In the following code, it is assumed that len1 (respectively len2) is equal to the length of the buffer buf1 (respectively buf2). (Note that this question has multiple correct answers.)

   ```
   void func( char *buf1, *buf2, unsigned int len1, len2) {
        char temp[256];
        if (len1 > 256) {return -1;}
        memcpy(temp, buf1, len1);
   }
   ```

   Answer: No, the code as given cannot have a buffer overflow. However, if we change the "char temp[256]" declaration to

   ```
   char temp[25];
   ```

   then we can get a buffer overflow.

2. Use the local optimization discussed in class (algebraic simplification, dead code elimination, ...) to optimally simplify the code below:

   ```
   a := x^2
   b := 3
   c := x
   d := c * c
   e := b * 2
   f := a + d
   g := e * f
   ```

   Answer: The optimized code is:

   ```
    a := x * x
   f := a + a
    g := 6 * f
   ```

3. We discussed the graph coloring algorithm in the context of the register allocation problem. It is known that graph coloring is an NP-hard problem. However, in many cases the following heuristic works well. Given a graph G and k colors, we determine if G is k-colorable as follows:

- Pick a node t with fewer than k neighbors in RIG
- Eliminate t and its edges from RIG
- If resulting graph is k-colorable, then so is the original graph

Why can we conclude that if after steps 1,2 above the resulting graph is k-colorable, then so is the original?

Answer: Let $c_1, ..., c_n$ be the colors assigned to the neighbors of t. Since $n < k$ we can pick some color for t that is different from those of its neighbors.

4. In the following piece of code is it possible to perform global constant propagation without altering its behavior. Explain your answer.

```
X := 3;
if( B + X == 7) {
X := B;
Y := f(X);
}
else{
X := 4;
Y := f(X);
}
return Y;
```

Answer: Yes. The simplified code is "return f(4);"

5. Short questions on bottom-up and top-down parsing

   (a) Briefly describe why bottom-up parsers are typically considered as more powerful than top-down parsers.

   (b) Does the following grammar, as given, have a recursive-descent parser?

$$S \to Sa \mid \epsilon$$

Answer: The answers are:

   (a) For bottom-up parsers, the grammar doesn't need to left-factored, and supports a larger class of CFLs.

   (b) No, the above-mentioned grammar has a left recursion.

6. The following grammar has left recursion. Rewrite the grammar such that it accepts the same language, but doesn't have left recursion.

$$A \to Aa \mid b$$

Answer: The idea is to introduce a new non-terminal, say, C and rewrite the rules appropriately:

$$A \to bC$$
$$C \to \epsilon \mid aC$$

Question 3: Global Constant Propagation Analysis (20 points)

In class we studied the global constant propagation algorithm. This algorithm has many interesting features. First of all, it uses labels to label the incoming and outgoing edges of each statement in the control-flow graph of a program (procedure is described below). The labels refer to a specific variable (say, x), and indicate whether the variable is a constant (c), not a constant (top), or undetermined (z).

Recall the control-flow graph of a program, where nodes are basic blocks and edges are control-flow edges between them. For every program variable X, for each edge of the control-flow graph, two labels are placed, one at the tail of the edge and the other at the head of the edge. The label at the tail of an edge is called an "out-label". The label at the head of an edge is called an "in-label". (An alternate way to think about this is that there is a label (in-label) before a statement S, and a label (out-label) after a statement S.)

We discussed several rules to compute the value of a variable X in an in-label of a statement S, using the out-labels on the edges incoming to S. We similarly discussed rules to compute the value of a variable X in an out-label after a statement S, by combining the value of X in an in-label and the semantics of the statement S.

1. Briefly describe how the presence of loops can complicate the computation of values of variables in in- and out-labels. Use an example.

   Answer: The trouble with loops is that in order to compute the out-labels of a loop's basic block (for simplicity, assume that the loop-body is a single basic block), we have to know the value of the variables in the in-labels. However, since the out-label of the loop-body is also the in-label for the loop's entry, this can cause difficulty for the global copy propagation alogrithm.

   This problem is resolved using two modifications to the basic global copy propagation algorithm:

   First, labels of the form "Label: X = z", which means that X is undetermined at that point in the code, since the analysis has not yet determined if control reaches that point.

   Second, The rules for computing in and out-labels are appropriately modified to handle the "X = z" case, and a partial order over label-values for variables is defined. The partial order is

   $$top$$

   $$... - 2, -1, 0, 1, 2, ...$$

   $$z$$

   The rules are updated to handle the X = z case such that the value of variable in an out-label is the least upper bound on all the values of the variables in the corresponding in-labels.

   We will discuss the example from the lecture notes.

2. You may be given an example global constant propagation with an incorrectly computed label. You will have to identify and fix the appropriate label.

   Answer: We will discuss the example from the lecture notes with appropriate modifications.

Question 4: Register Allocation (30 points)

Answer: I will review this material in class, and ask some questions based on it.