COP4020
Programming Language Concepts
Spring 2017
Sample Final Exam Questions

Disclaimer: This is a collection of sample exam questions, not a sample exam.
1. Add the unary postfix auto-increment operator ‘++’ to the RPAL Tiny interpreter. The AST is the form (‘post++’, ‘n’). To evaluate this operator, first evaluate the operand ‘n’. Then increment ‘n’ (changing memory). The resulting value is the value of ‘n’ before the increment.

2. Make the ‘else’ clause optional in the ‘if’ statement in the RPAL Tiny interpreter.

3. Add a ‘read’ statement to the RPAL Tiny interpreter. This is in addition to the existing ‘read’ expression. The AST is of the form (‘read’, ‘n’). To execute this statement, extract a value from the input, and store it in variable ‘n’ (changing memory).

4. Add the ‘repeat’ statement to the RPAL Tiny interpreter. The AST is of the form (‘repeat’, S, E). The statement S is executed at least once. After that, E is evaluated: if false, S is executed and E is re-evaluated; if true, the iteration stops.

5. Add the for statement (as in C, C++ or Java) to the RPAL Tiny interpreter. The AST is of the form (‘for’, S1, E2, S3, S4), where S1, S3, and S4 are statements, and E2 is an expression. Hint: (‘for’, S1, E2, S3, S4) is equivalent to (‘;’, S1, (‘while’, E2, (‘;’, S4, S3)))).

Note: On the final exam, for questions such as these (1-5), the Tiny RPAL interpreter will be provided.

6. Write a complete RPAL program that defines and uses a function to merge two strings. Assume the characters in each string are in ascending order. For example,

   merge ‘acegh’ ‘bdfi’ = ‘abcdefghi’
7. Write a complete RPAL program that defines and uses a function to remove the last n characters from a given string, if possible. For example,

```
remove ‘abcdef’ 3 = ‘abc’
```

8. A severe system crash takes place on your computer system, leaving you with an implementation of a Pascal-like language in executable form. The source code is gone forever. You remember that before the crash there were three executable compilers. One used copy-in,copy-out as the parameter passing mechanism; another other used pass by reference, and the third used pass-by-value. You don't know which compiler survived. The only difference between the three compilers was the parameter passing mechanism. Write a (short) program in Pascal/C-like pseudo-code that will allow you to determine which compiler survived the crash. Specifically, write a program whose output will be 0 if copy-in,copy-out is used, 1 if pass by reference is used, and 2 if pass-by-value is used. Show a legible trace of the program's execution.

9. For the program shown below, draw a picture of the run-time environment, at the point marked "HERE", i.e., at the point when the constant value “1” has been placed on the stack. Show all temporaries, local variables, parameters, locations reserved for return values, and return addresses, as well as the base pointer, frame pointer(s), and the stack pointer. Use the frame organization illustrated in Lecture 12.2.

```
function fact(int n) {
    int t;
    if n = 0 then return 1  // HERE
    else {
        t=fact(n-1);
        return n*t;
    }
}
main()
begin
    write(fact(3))
end;
```
10. Given the “family relation” facts shown in the next-to-last example in Lecture 14.3 (defining FatherOf and MotherOf predicates), write a predicate CousinOf(X,Y). Trace the search process through the database, to answer the query ?-CousinOf(ken,X). Explain why other family relations, such as UncleOf(X,Y) and Niece(X,Y), are unrealistic with the given database.