

CS 405/505 PROGRAMMING LANGUAGES
TOPICS TO BE COVERED ON FINAL EXAM
December 11, 2007

The exam will cover *everything* we have discussed since the second exam material ended. This includes topics from Chapters 5-12 in the text, but emphasizing the material covered in class. Detailed questions will not be asked about topics covered on the first two exams but you should be aware of the central topics (e.g. the structure of a compiler, the importance of formal semantics, etc.). The exam should last approximately 2.5 hours. The style of the exam will be very flexible, possibly consisting of fill in the blank, true or false (possibly with justification), multiple choice, matching, short answer (e.g. definitions or listing), and discussion questions. You will not be asked to work detailed problems such as writing any actual code but you should have some ideas about the theory (e.g. what really does happen in procedure calls). Some smaller problems such as illustrating the difference between static scope and dynamic scope or managing a stack of activation records are likely. In general, you should know how to solve the problems given in assignments even though you may not have to actually solve such problems on the exam.

A brief outline of the topics we have covered is described below. (This list is intended to be as complete as possible but may not be all inclusive.)

- *Data Types.* The key concepts were aliasing, static vs. dynamic binding, particularly of types, static vs. dynamic scope, and the notion of environment. Among structured data types, we concentrated on arrays, records, unions, and pointers, including garbage collection.
- *Expressions.* The key concepts were the operand evaluation order, including short-circuit Boolean expressions, functional side effects, operator overloading, and type coercion.
- *Statements.* The design and implementation of conditional and looping statements were discussed, including if, while, switch, and for types of statements, and guarded commands.
- *Subprograms.* The management of subprogram activations was discussed for the simple call-return model, including recursive subprograms. Central issues were the implementation of non-local referencing environments, parameter-passing methods, aliasing, static vs. dynamic scope, static chain vs. display, shallow vs. deep binding, and shallow vs. deep access.
- *Data Abstraction and Object-Oriented Programming.* C++ was introduced as an object-oriented programming language and contrasted with Java. Generic units in Ada and C++ were discussed along with static vs. dynamic binding.

CS 405/505 FINAL EXAM SAMPLE QUESTIONS

1. True or False. A virtual method table is needed only for the execution of polymorphic functions in C++, but is not needed in Java.
2. List and briefly describe four (4) components of an activation record.
3. Consider the Pascal array declaration:

A : array [-10 .. 10, -5 .. 5, 1 .. 10] of T;

Assuming that T requires 10 bytes of storage and A is stored at relative location 100, what is the relative location of A [1, 1, 2]?

4. Define the terms *dangling reference*, *garbage*, and *memory leakage*. Describe how each is created in a program and a possible solution to each.
5. Consider the following Pascal program:

```
program MAIN;
  var U, V : INTEGER;
  procedure A;
    var Y : INTEGER;
    ... (* point 4 *)
  end;
  procedure B;
    var U, V, Y : INTEGER;
    procedure C;
      ...
      begin
        ... (* point 3 *)
        A;
        ...
      end;
    begin (* B *)
      ... (* point 2 *)
      C;
      ...
    end;
  begin (* MAIN *)
    ... (* point 1 *)
    B;
    ...
  end.
```

Indicate which activation records are stacked at each of points 1, 2, 3 and 4 in the program as well as the exact contents of those activation records. Assuming static scoping, show the contents of the display at each of these points. Assuming dynamic scoping, what is the scope of V at point 4?

6. Consider the functions below in C++-like pseudocode.

```
int I;

int F (int X, int Y) {
    I = 2;
    Y = 1;
    return X + Y;
}

void main (void) {
    int A [3];
    A [0] = 7; A [1] = 11; A [2] = 13;
    I = 0;
    cout << F (A [I], I);
}
```

Show the values stored in the **locations** named A [0], A [1], A [2], I, X and Y at the entry and exit points of function F assuming (a) call by value, (b) call by reference, (c) call by value-result, and (d) call by name? Also, show the result which gets printed in each case.

Assuming all parameters are passed by reference, does *aliasing* occur in this program? If so, where?