

CS 405 PROGRAMMING LANGUAGES
TOPICS TO BE COVERED ON FINAL EXAM
December 11, 2003

The exam will cover *everything* we have discussed since the second exam material ended through Thursday, December 5. This includes topics from Chapters 3, 5-10, and 16 in the text and all hand-outs, 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.)

- *Programming Languages - Semantics.* We discussed dynamic semantics and formal methods to describe it. Of the three formal methods, operational, denotational and axiomatic, we concentrated on denotational and axiomatic. You should know the goals of formal semantics and should have specific ideas about how denotational semantics may model interpretation of programs and how axiomatic semantics may be used in program verification. You need not remember any specific denotational semantics equations or axiomatic rules per se but should know enough about them to be able to explain or use them if they are given to you.
- *Denotational Semantics.* You should know the principles of this from the standpoint of program interpretation. You need not remember any denotational rules per se but should know enough about them to be able to explain or use them if they are given to you.
- *Prolog and Logic Programming.* You should know the general design philosophy and syntactic and semantic flavor of Prolog. You do NOT need to know specific commands. Examples of the most detailed information of a fundamental nature you should know would be how the rule matching works (e.g. execution of rules, instantiation of variables, backtracking, etc.) and the representation of lists.
- *Axiomatic Semantics.* You should know the principles of this from the standpoint of program verification. You need not remember any axiomatic rules per se but should know enough about them to be able to explain or use them if they are given to you.
- *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, 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.

CS 405 FINAL EXAM SAMPLE QUESTIONS

1. $A(n)$ ----- is a condition which is true upon entry to a loop every time the loop is entered, and is still true at the termination of the loop.
2. Consider the Prolog program which solves the Towers of Hanoi problem. This problem is to move a stack of blocks from one location to another without ever changing the order in which blocks are placed on top of each other. A third stack may be used but only one block may be moved at a time.

```
hanoi(N) :- move(N, left, center, right).
```

```
move(0, X, Y, Z).
```

```
move(N, X, Y, Z) :-
```

```
    N > 0, M is N - 1, move(M, X, Z, Y), print([X,Y]), move(M, Z, Y, X).
```

Trace the above program using the query `hanoi(3)`. Show all queries which are generated by the execution and all moves which are printed. You may wish to draw a tree to show this clearly.

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. Consider the denotational semantics below:

$$S[\mathbf{S}_1; \mathbf{S}_2] \text{ store} = S[\mathbf{S}_2] (S[\mathbf{S}_1] \text{ store})$$

$$S[\mathbf{V} := \mathbf{E}] \text{ store} = \text{store}[E[\mathbf{E}] \text{ store} / \mathbf{V}]$$

$$S[\mathbf{while} \mathbf{C} \text{ loop } \mathbf{S} \text{ end loop}] \text{ store} =$$

$$\text{if } C[\mathbf{C}] \text{ store then } S[\mathbf{while} \mathbf{C} \text{ loop } \mathbf{S} \text{ end loop}] (S[\mathbf{S}] \text{ store}) \text{ else store}$$

$$C[\mathbf{E}_1 > \mathbf{E}_2] \text{ store} = \text{if } E[\mathbf{E}_1] \text{ store} > E[\mathbf{E}_2] \text{ store then true else false}$$

$$E[\mathbf{E}_1 + \mathbf{E}_2] \text{ store} = E[\mathbf{E}_1] \text{ store} + E[\mathbf{E}_2] \text{ store}$$

$$E[\mathbf{E}_1 - \mathbf{E}_2] \text{ store} = E[\mathbf{E}_1] \text{ store} - E[\mathbf{E}_2] \text{ store}$$

$$E[\mathbf{I}] \text{ store} = N[\mathbf{I}]$$

$$E[\mathbf{V}] \text{ store} = \text{if store}[\mathbf{V}] = \perp \text{ then } \top \text{ else store}[\mathbf{V}]$$

Assume that `N` returns the integer value of its argument. Given the initial store $(\lambda \mathbf{V}.\perp) [1/u] [5/y] [5/z]$, compute the store which results from evaluating `S` on the program below, showing all steps used in the evaluation.

```
while (u > 0) loop
  u := u - 1;
  z := z + y
end loop
```

5. Consider the axioms of assignment, composition, and loop below:

Assignment $\{P [E/V]\} V := E \{P\}$

Composition

$$\frac{\{P\}S_1\{Q\}, \{Q\}S_2\{R\}}{\{P\}S_1; S_2\{R\}}$$

Loop

$$\frac{\{P \& B\}S\{P\}}{\{P\}\text{while } B \text{ loop } S \text{ end loop}\{P \& \neg B\}}$$

Prove the correctness of the following program segment.

```
{n = j * (j + 1) / 2 & i >= 0}
while j != i loop j := j + 1; n := n + j; end loop
{n = j * (j + 1) / 2 & i >= 0 & ¬(j! = i)}
```

Show all steps used in the proof.

6. Consider the following Pascal program:

```
program MAIN;
  var U, V : INTEGER;
  procedure A;
    var Y : INTEGER;
    ... (* point 3 *)
  end;
  procedure B (procedure X);
    var U, V, Y : INTEGER;
    procedure C;
      ...
      begin
        ... (* point 4 *)
        Y := ...;
        ...
      end;
    begin (* B *)
      ... (* point 2 *)
      X;
      B (C);
      ...
    end;
  begin (* MAIN *)
    ... (* point 1 *)
    B (A);
    ...
  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 and deep binding, show the contents of the display at each of these points. Assuming dynamic scoping, what is the scope of Y at point 4?

7. 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?