MIDTERM EXAMINATION

CSE 130: Principles of Programming Languages

Professor Goguen

February 16, 2006

---

100 points total

- *Don't start the exam until you are told to. Turn off any cell phone or pager.*
- *Write your name and PID in the top-right corner of this page. Do not separate the pages.*
- *This is a closed-book, closed-notes, no-calculator exam. Don't refer to any materials other than the exam itself.*
- *Do not look at anyone else's exam. Do not talk to anyone but an exam proctor during the exam.*
- *If you have a question, raise your hand and an exam proctor will come to you.*
- *You have 80 minutes to finish the exam. When time is up, you must stop writing.*

Do not turn this page until you are told that you may do so

History and Culture of Programming Languages

1. [2 points] A main point of the *Essay on Comparative Programming Linguistics* is:

a. Comparing programming languages to one another is more fruitful than studying one in isolation.

b. The study of programming languages should employ tools from linguistics such as Chomsky's formal grammars.

c. In order to understand programming languages in depth, one must study the context of their use.

d. We should adopt a paradigmatic rather than syntagmatic focus when discussing the merits of programming languages.

2. [2 points] The first high-level imperative computer programming language was

a. Algol 60.

b. Lisp.

c. Pascal.

d. Fortran.

3. [2 points] The *functional programming paradigm* grew out of

a. the theory of Turing machines.

b. the lambda calculus.

c. need to overcome the limitations of early imperative languages.

d. early artificial-intelligence research at MIT.

Expressions, Grammars, and Syntax

4. [4 points] Convert the infix expression $b * b - 4 * a * c$ to both prefix and postfix notation. Assume the standard precedence of operators.

Prefix: ____

Postfix: ____

5. [20 points] Use the following grammar for parts a–c. S is the start symbol, italicized capital letters (S, T, V) denote non-terminal symbols, and other letters, numbers, and symbols denote terminal symbols.

$$
\begin{array}{lcl}
S & ::= & \lambda\, T \,\textbf{.}\, S \\
S & ::= & (\; S\; S\;) \\
S & ::= & T | V \\
T & ::= & \mathbf{w}|\mathbf{x}|\mathbf{y}|\mathbf{z} \\
V & ::= & \mathbf{1}|\mathbf{2}|\mathbf{3}
\end{array}
$$

a. [5 points] Are the following strings in the language of the grammar?

λ **w . λ x . (w x)**	Yes	No
((x x) x) (y y)	Yes	No
λ **x 2**	Yes	No
λ **x . (2 λ x . y)**	Yes	No
λ **(x x) . (x x)**	Yes	No

b. [5 points] Draw a parse tree for the following string. If there are multiple possible parse trees, any are acceptable.

λ **x . (λ y . λ z . (x y) z)**

c. [10 points] Is the grammar ambiguous? Why or why not?

6. [10 points] Convert the following grammar to an equivalent unambiguous grammar using the template provided. S is the start symbol. Both $=$ and $\hat{}$ should be right associative; but $+$, $-$, and $*$ should be left associative. The $=$ should have the lowest precedence, followed by $+$ and $-$ (second lowest), then $*$ (second-highest) and $\hat{}$ (highest).

$$
\begin{array}{lcl}
S & ::= & S = S \\
S & ::= & S \,\hat{}\, S | S + S | S - S | S * S \\
S & ::= & \mathbf{0}|\mathbf{1}|\mathbf{2}|\mathbf{3} \\
S & ::= & \mathbf{w}|\mathbf{x}|\mathbf{y}|\mathbf{z}
\end{array}
$$

Use the following template for your unambiguous grammar:

S ::=

T ::=

U ::=

V ::=

W ::=

Structured Programming and Invariants

7. [2 points] Advocates of structured programming argued that:

a. Programming languages should support the clean division of problems into smaller units.

b. Programmers should structure their code by writing invariants first.

c. Encapsulation is important in the design of large software systems.

d. A program's control flow should be evident from its syntactic structure.

8. [10 points] The following program fragment rearranges the elements of an array $A[1..n]$ into non-decreasing order using a variant of selection sort.

A precondition, a postcondition, and several invariants have been provided for you, but important invariants are missing from the 2 blank spaces. In these spaces write thorough, precise mathematical invariants that logically follow from preceding invariants and imply those following.

$\{n \geqslant 1\}$
$b := 1;$
while $b < n$ **do**
    begin
                                                            $\longleftarrow$ missing invariant

    $m := b;$
    while $m < n$ **do**
        begin
        $m := m + 1;$
        $\{\forall i : b \leqslant i < m \rightarrow A[b] \leqslant A[i]\}$
        if $A[m] < A[b]$ **then**
            begin
            $t := A[b];$
            $A[b] := A[m];$
            $A[m] := t;$
            end
        $\{A[b] \leqslant A[m]\}$
                                                            $\longleftarrow$ missing invariant

        end
    $\{m = n\}$
    $\{\forall i : b < i \leqslant n \rightarrow A[b] \leqslant A[i]\}$
    $b := b + 1;$
    end
$\{b = n\}$
$\{\forall i, j : 1 \leqslant i \leqslant j \leqslant n \rightarrow A[i] \leqslant A[j]\}$

Types and Type Equivalence

9. [15 points] Consider the following type definitions.

type $A =$ **array** $[0..9]$ **of integer**
type $B =$ **array** $[0..9]$ **of integer**
type $C = A$
type $D = \uparrow A$
type $E = D$

In each cell of the grid below, write

- S if the types in the row and column headers are structurally equivalent,
- T if they are transitive-name equivalent, and
- P if they are pure-name equivalent.

If two types are equivalent under all 3 kinds of equivalence, you should write STP; if they are not equivalent under any, you should leave the cell blank.

	A	B	C	D	E
A					
B					
C					
D					
E					

10. [5 points] Briefly state the difference between strongly typed and weakly typed languages.

Parameter Passing and Scope

11. [20 points] Consider the following Pascal code:

procedure $P(x, y : integer, z : \uparrow integer) : integer$
  var $c : \uparrow integer;$
begin
  new$(c);$
  $c\uparrow := 0;$
  $x := x + 1;$
  $z := c;$
  $y := y + 1;$
  $P := z\uparrow$ (* This is the same as **return** $z\uparrow$ *)
end

var $a, c : \uparrow integer;$
var $b : integer;$
new$(c);$
$a := c;$
$c\uparrow := 4;$
$b := P(c\uparrow, c\uparrow, c);$

For each parameter-passing method in the table below, fill in the values of $a\uparrow$ and b after running the Pascal code above.

	$a\uparrow$	b
Call-by-Value		
Call-by-Value-Result		
Call-by-Reference		
Call-by-Macro Expansion		
Call-by-Name		

12. [8 points] Briefly state what tail recursion is and why it is important.