I have neither given nor received unauthorized aid on this examination, nor do I have reason to believe that anyone else has.

__________________________
Name (printed)

__________________________
Signature

CS 355
Principles of Programming Languages
Midterm Exam
Fall 2003

Instructions: This is a closed-book, closed-notes exam. All work is to be done on these pages. If you need more room for a particular question than is provided, use the backs of these pages but be sure to label your answers clearly. There are a total of 50 points worth of questions on this test. The point values assigned to each question represent a rough indicator of the difficulty and/or level of detail expected. Use your time accordingly.

Your answers, especially where judgements or explanations are requested, should be precise and to the point. Completeness and precision are important, but extraneous material will not be rewarded and may actually cost you points if it gives me the impression that you do not know what part of your own answer is actually important. Remember, it is up to you to demonstrate that you know the material – it is not up to me to try to ferret it out.
1. (8 pts) Give a regular expression for variable names according to the following rules:

   (a) A variable name consists of one or more characters.
   • Every variable name must begin with an alphabetic character or an underscore ("_").
   • After the first character, the remaining characters of a variable name must be alphabetic, underscores, or numeric digits.

   You may employ any of the extensions to the basic regular expression syntax that were described in class.

   

   \[[A-Za-z][A-Za-z0-9]^{\ast}\]

   (b) Give a regular expression for variable names that must follow the preceding rules plus the following:

   • A variable name may not contain two or more underscores in immediate succession.
   • A variable name may not consist solely of an underscore. It must contain at least one alphanumeric character.

   \(([A-Za-z]([-A-Za-z0-9]) ([A-Za-z0-9]([-A-Za-z0-9])^{\ast})^{\?}\)

   or

   \((([A-Za-z]([-A-Za-z0-9]) ([A-Za-z0-9]([-A-Za-z0-9])^{\ast})^{\?}\)

2. (7 pts) Give a BNF grammar for function calls in a C++-like language, where a function call begins with a function name followed by a parenthesized list of zero or more expressions separated by commas. For the expressions, use the nonterminal \langle exp \rangle and do not give any productions for \langle exp \rangle.

   \langle call \rangle ::= \langle id \rangle (\langle paramList0 \rangle )
   \langle paramList0 \rangle ::= | \langle paramList1 \rangle
   \langle paramList1 \rangle ::= \langle exp \rangle | \langle exp \rangle , \langle paramList1 \rangle

3. (10 pts) Given the following grammar

   \langle stmt \rangle ::= \langle loop \rangle | s
   \langle loop \rangle ::= \langle while \rangle | \langle repeat \rangle
   \langle while \rangle ::= while \langle exp \rangle do \langle stmt \rangle
   \langle repeat \rangle ::= repeat \langle stmt \rangle until \langle exp \rangle
   \langle exp \rangle ::= c
(a) Draw the parse tree for
while \( c_1 \) do repeat \( s \) until \( c_2 \)

(b) Draw the corresponding abstract syntax tree for that same statement.
4. (6 pts) Consider the following construct:

\[
\text{while } (\text{cond}) \text{ do } \text{SL} \text{ end}
\]

where \text{cond} represents some arbitrary boolean expression and \text{SL} is some list of statements. Show how this statement would be translated into a “pseudo-assembly language” consisting of statements of the form:

\[
\text{goto } \langle \text{Label} \rangle \\
\text{if } \langle \text{var} \rangle \text{ is zero then goto } \langle \text{Label} \rangle \\
\text{if } \langle \text{var} \rangle \text{ is nonzero then goto } \langle \text{Label} \rangle \\
\langle \text{var} \rangle_1 = \langle \text{var} \rangle_2 \op \langle \text{var} \rangle_3 \\
\langle \text{var} \rangle_1 = \langle \text{op} \rangle \langle \text{var} \rangle_2
\]

where any statement can be preceded with a \text{Label}, e.g.,

L1: if X is zero goto L2;

and where \langle \text{op} \rangle could be any one of: + -.

You need not provide a translation of \text{cond} and \text{SL}, but should indicate where within your translation the code for them would appear. Assume that the translator processes code in one pass from start to finish and can only “buffer” or “remember” a finite amount of translated code before writing it out, i.e., the code for \text{cond} must appear before the code for \text{SL}.

Any other assumptions required for your translation should be stated explicitly.

Assume that the translation of \text{cond} leaves a boolean value in C.

L1: translation of \text{cond} \\
if C is zero go to L2; \\
translation of \text{SL} \\
goto L1 \\
L2:

5. (9 pts)

(a) What is strong typing?

The property of a language that guarantees that it accepts only safe expressions (expressions that can be evaluated without a type error).

(b) What is type coercion?

The automatic introduction by the compiler of type conversion code in circumstances that would otherwise indicate a type error.

(c) What is overloading?

Having multiple functions or operators with the same name but different signatures.
6. (4 pts) A certain programming language allows programmers to assign two records (structs) to one another provided that the two have the same number of fields (data members), and each corresponding pair of fields from the two records have the same type.

This is evidence that the type system is based on...?

structural equivalence

7. You are writing some code to manipulate a data type named “Foo”. The designer of “Foo” is a paranoid schizophrenic who is afraid that everyone is out to steal his code. He would only allow you one quick look at its declaration:

```c++
??? Foo {
    int foo1;
    double foo2;
    char foo3;
};
```

Foo x;

Unfortunately, he flashed the printout in front of you so quickly that you didn’t have a chance to check whether the “???” was the word “struct” or “union”.

He then disappeared into his cubicle and refuses to speak to you. Due to a strange encryption scheme (did I mention that the Foo designer is paranoid?), you are able to compile with the Foo declaration, but cannot read the file where it resides.

(a) (3 pts) Write a simple C++ expression that uses `sizeof()` and can be plugged into the following code to determine the unknown structure of Foo:

```c++
if (place-your-expression-here)
    cout << "Foo is a struct" << endl;
else
    cout << "Foo is a union" << endl;

if (sizeof(Foo) >= sizeof(int) + sizeof(double) + sizeof(char))
```

A union is the size of its largest member. The size of a struct is the sum of the sizes of its members, plus possible padding.

(b) (3 pts) It’s just not your day. Someone has disabled the `sizeof()` function. Write another test expression to determine the structure of Foo, this time using the `&` “address-of” operator.

---

1. `sizeof(T)`, where T is a type, returns the number of bytes occupied by an object of that type when used in an array.
if ((char*)(&(x.foo1)) != (char*)(&(x.foo2)))

In a union, all the members start at the same address. In a struct, each member follows the other, at a higher address.