Name:
SSN :

CSE 428

Fall 1999

Midterm #1

6 October 1999

The exam consists of 5 problems on 5 pages, totaling 100 points. Read each question carefully and use your time judiciously.

Write your name/number on every page.

1. Consider the following grammar: (20 pts)

$$\begin{aligned}
E &::= E\text{``}+\text{''}F \mid E\text{``}-\text{''}F \mid F \\
F &::= F\text{``}*\text{''}F \mid G \\
G &::= \text{``}-\text{''}G \mid H \\
H &::= N \mid Id \mid \text{``(''}E\text{``)''}
\end{aligned}$$

in which N and Id are nonterminals which unambiguously generate a non-empty sequence of digits and identifiers beginning with an alphabetic character, respectively.

For each of the following sentences, state whether it can be generated by the grammar, and if it can, whether it can be generated unambiguously.

(a) $x * (y - -5)$

(b) $+x * y$

(c) $x * y + z * v$

(d) $(x + z) * -u * (v)$

(e) $-(-(x + y))$

2. Recall the typing of a single recursive function declaration: (20 pts)

$$\frac{senv[f : \tau_1 \to \tau_2,\, x : \tau_1] \vdash e_2 : \tau_2 \quad senv[f : \tau_1 \to \tau_2] \vdash e : \tau}{senv \vdash \texttt{let}\ f(x : \tau_1) : \tau_2 = e_2\ \texttt{in}\ e\ \texttt{end}\ :\ \tau}$$

Consider the extension of the language to support exactly 2 mutually recursive function declarations. We might write such a declaration as:

```
let f(x:τ1):τ2 = ef
    g(y:τ3):τ4 = eg
in e
end
```

where `f` and `g` can appear in `e`, $\texttt{e}_\texttt{f}$ and $\texttt{e}_\texttt{g}$.

Give a typing rule, similar to the one above, to type such expressions.

3. Using the typechecking rules given in class (and on assignments), type each of the following expressions. Assume sequences of declarations in one `let` expression are handled *sequentially.* If the expression is not well-typed, then write **no type**; otherwise, give the type of the expression. (20 pts)

(a)
```
let f(x:int):int = x*2;
    g(x:int):bool = x>0;
    z = 5
in  g(f(z+2)) and g(f(3+f(z)))
endlet
```

(b)
```
let x = 23;
    y = x * x;
in  let z = x*y;
        y = x>y
    in y+z
    endlet
endlet
```

(c)
```
let sq(x:int):int = x*x;
    qd(y:int):int = sq(sq(y));
    z = true
in  (let z=qd(4) in sq(z) endlet) + sq(4)
endlet
endlet
```

(d)
```
let x = (let x = 5 in x>0 endlet)
in let x = (if x then 3 else 6)
   in let x = x*x
      in x>0
      endlet
   endlet
endlet
```

4. Consider the following program: (20 pts)

```
program main
  x,y : integer;

procedure petruchio(a : integer, b : integer)
begin
  y := 43;
  a := b;
  b := 7;
  write(a,b);
end petruchio;

begin main
  x := 1;
  y := 0;
  petruchio(x,y);
  write(x,y);
end main;
```

Assuming the language is statically scoped, give the output of this program when the parameters are passed

(a) call-by-value

(b) call-by-value-result

(c) call-by-reference

(d) call-by-need

5. Which of the following language restrictions **allow** us to implement a language **without** static links or displays? (Write **yes** for each item which does allow this, write **no** otherwise.) (20 pts)

(a) procedures can only reference their parameters and local variables

(b) dynamic typing

(c) static scoping

(d) only call-by-value parameter passing

(e) no recursive functions

(f) no nested procedure declarations

(g) static typing

(h) no overloading of procedure names

(i) dynamic scoping

(j) recursive procedures and functions must be tail-recursive.