Login name _____ **Quiz 4** **Name** _____

CSE 131B

Signature _____ **Spring 2006** **Student ID** _____

1. Write the Oberon program (in the box to the right) that corresponds to the following SPARC assembly and state what the program outputs.

```
        .section    ".text"
        .align      4
        .global     foo
foo:
        set         foo.SIZE, %g6
        save        %sp, %g6, %sp

        ! Real parameter "x" is in %f0
        fadds       %f0, %f0, %f0

        ret
        restore

foo.SIZE = -(92 + 0) & -8

!-----------------------------------

        .section    ".data"
        .align      4
tempr:  .single     420.25

        .section    ".text"
        .align      4

        .global     main
main:
        save        %sp, -96, %sp

        set         tempr, %l0
        ld          [%l0], %f0
        call        foo
        nop

        call        printREAL
        nop

        mov         %g0, %i0

        ret
        restore
```

What is the output of this program? _____

2. Which part of a function call is typically responsible for saving the current value of the program counter as the return address (caller or callee)? _____.
And which SPARC instruction actually accomplishes this? _____

3. What is arguably the biggest disadvantage of macros as an open subroutine compared to closed and leaf subroutines (meaning this disadvantage is not present in closed and leaf subroutines)?

4. Consider the following Oberon program fragment.

```
VAR x : INTEGER;           (* assume x is located at %g7 - 4 *)

FUNCTION fubar (a : INTEGER) : INTEGER;
       BEGIN
              ...
              RETURN 0;
       END fubar;

FUNCTION snafu (REF a : INTEGER) : INTEGER;
       BEGIN
              ...
              RETURN 0;
       END snafu;

FUNCTION foo (a : INTEGER; REF b : INTEGER) : INTEGER;
       VAR y : INTEGER;    (* assume y is located at %fp - 4 *)
       BEGIN
              (* XXXXX *)
              RETURN 0;
       END foo;
```

If you make different function calls to fubar() and snafu() at the point in function foo() marked XXXXX, different SPARC assembly instructions are needed to correctly pass the different actual arguments to these functions. For example, if at the line marked XXXXX in foo() you were to call fubar(x); you would need to generate the correct instruction(s) to pass the actual argument x to fubar() matching the parameter passing mode specified in the definition of fubar().

The possible instructions you are allowed to use are:

```
A) ld    [%g7 - 4], %o0          I) st    %i0, [%fp + 68]
B) add   %g7, -4, %o0            J) add   %fp, +68, %o0
C) ld    [%fp - 4], %o0          K) st    %i1, [%fp + 72]
D) add   %fp, -4, %o0            L) add   %fp, +72, %o0
E) mov   %i0, %o0                M) st    %i0, [%fp - 4]
F) mov   %i1, %o0                N) st    %i0, [%g7 - 4]
G) ld    [%i0], %o0              O) st    %i1, [%fp - 4]
H) ld    [%i1], %o0              P) st    %i1, [%g7 - 4]
```

Specify the letter(s) matching the instruction(s) needed to correctly pass the argument in the following function calls. If more than one instruction is needed, specify the letters in the correct order (for example, if the instruction tagged with the letter N should be executed first followed by the instruction tagged with the letter D, your answer should be ND). Note there are no loads from %fp+68 or %fp+72.

```
fubar(x);   _______          snafu(x);   _______

fubar(y);   _______          snafu(y);   _______

fubar(a);   _______          snafu(a);   _______

fubar(b);   _______          snafu(b);   _______
```