Signature_____ **Quiz 5** **Name** _____

CSE 131

Login name _____ **Spring 2008** **Student ID** _____

1. Given the following Reduced-C code fragment:

Reduced-C

```
function : int foo( int & a, int b )
{
  return a | b;
}
```

Complete the SPARC Assembly language statements that might be emitted by a compliant Reduced-C compiler from this quarter for function foo().

```
        .section _________
        .global _________
        .align 4
foo:
        _______ FOO_SAVE, %g1
        save    ________, %g1, ________

        st      _______, [%fp + 68]     ! Store the 2 params in the proper
        st      %i1, [____________]     ! param locations in stack frame

        _______ [%fp + 68], %o0         ! Get value of object referenced
        ld      [__________], %o0       ! by param 1

        ld      [__________], %o1       ! Get value of param 2

        _______ %o0, %o1, %o0           ! Perform bitwise OR
        st      %o0, [__________]       ! Store result in tmp1 on stack

        ld      [__________], %o0       ! Get result of previous expr (yes, redundant ld)
        mov     %o0, ________           ! Put in return value register

        ret                             ! return
        ___________

        FOO_SAVE = -(_______ + 4) & -8  ! Save space for 1 tmp/result on stack
```

2. With regard to saving registers in the calling convention, why is callee-save usually more efficient than caller-save?

In the Java run time environment, where are static variables located?

In the C/C++ run time environment, where are uninitialized internal (local) static variables located?

3. What gets printed in the following program?

Reduced-C

```
int a = 17;
int b = 71;

function : void fubar( int x, int & y, int p, int & q )
{
   x = x + 1;
   y = y + 1;
   p = p + 1;
   q = q + 1;
}

function : void foo1( int & c, int d )
{
   c = c + 1;
   d = d + 1;

   cout << a << endl;   _______
   cout << b << endl;   _______
   cout << c << endl;   _______
   cout << d << endl;   _______

   fubar( c, c, d, d );

   cout << a << endl;   _______
   cout << b << endl;   _______
   cout << c << endl;   _______
   cout << d << endl;   _______
}

function : int main()
{
   foo1( a, b );

   return 0;
}
```

What question would you like to see on the Final?