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

CSE 131

Login name _____ **Winter 2009** **Student ID** _____

1. Given the following Reduced-C code fragment:

Reduced-C

```
function : int * foo( int & a, int b )
{
  int x;
  x = a ^ b;
  b = x;
  return & b;
}
```

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

```
        .section __________

        .global  __________
        .align 4
foo:
        ________ FOO_SAVE, %g1
        save    _________, %g1, _________

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

        _______ [___________], %o0        ! Get value of object referenced
        ld      [___________], %o0        ! by param a

        ld      [___________], %o1        ! Get value of param b

        _______ _______, _______, %o0     ! Perform bitwise Exclusive OR (a ^ b)
        st      %o0, [%fp - 8]            ! Store result in a temp on stack

        ld      [%fp - 8], %o0            ! Load temp holding rhs value for assignment
        st      %o0, [___________]        ! Store result in local var x on stack (x = a ^ b;)

        _______ [___________], %o0        ! Get value of local var x on stack
        st      %o0, [___________]        ! Store this value into param b (b = x;)

        _______ _______, _______, %o0     ! Get address of b (&b)
        mov     %o0, _________            ! Put this in return value register

        ____________                      ! return from subroutine

        ____________

        FOO_SAVE = -(_______ + _______) & -8  ! Save space for 1 local var + 1 temp on stack
```

2. With regard to the above function/code, this will work most of the time, but there is a latent bug. What is it?

3. Name the part of the compilation sequence which performs the following.

_____ translates assembly code into machine code

_____ performs syntax analysis on its input high-level language (HLL)

_____ takes an executable file on disk and makes it ready to execute in memory

_____ combines all object modules into a single executable file

_____ translates HLL code (for example, C) into assembly code

_____ performs semantic analysis on its input high-level language (HLL)

_____ expands # directives & strips comments from its input high-level language (HLL)

_____ resolves undefined external symbols with defined global symbols in other modules

4. What is the 80/20 rule?

5. If there are 4 instances of a Foo type (4 objects of type Foo) as defined below:

```
class Foo
{
  private String s;
  private static Foo f;
}
```

How many copies of `f` are there in the Java run time environment? _____

Where is/are it/they located in the Java run time environment? _____

How many copies of `s` are there in the Java run time environment? _____

Where is/are it/they located in the Java run time environment? _____

What question would you like to see on the Final?