Login name _____   **Quiz 3**   **Name** _____

CSE 131

Signature _____   **Winter 2009**   **Student ID** _____

1. Given the following Reduced-C definitions:

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

float x;   /* global variables */
int y;
```

For each of the following statements, indicate the type of error (if any) that should be reported according to the Project I spec for this quarter. Use the letters associated with the available errors in the box below.

```
x = foo( foo( x ) );     ____

y = foo( x );            ____

x = foo( x + y );        ____

&x = foo( x );           ____

x = foo( 4.2 );          ____

x = foo( &x );           ____

x = foo( x );            ____

x = foo( y );            ____
```

> A) No Error
> B) Arg passed to reference param is not a modifiable L-val
> C) Argument not assignable to value param
> D) Argument not equivalent to reference param
> E) Left-hand operand is not assignable (not a mod L-val)
> F) Value of right-hand-side type not assignable to left-hand-side type

2. Show the memory layout of the following C struct/record definition taking into consideration the **SPARC** data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate struct/record member/field name. For example, if member/field name p takes 4 bytes, you will have 4 p's in the appropriate memory locations. If the member/field is an array, use the name followed by the index number. For example, some number of p[0]s, p[1]s, p[2]s, etc. Place an X in any bytes of padding. Structs and unions are padded so the total size is evenly divisible by the most strict alignment requirement of its members.

```
struct foo {
   char   a;
   int    b[2];
   double c;
   short  d[3];
   int    e
   char   f;
};

struct foo fubar;
```

What is the `offsetof( struct foo, c )`? _____

What is the `offsetof( struct foo, a )`? _____

What is the `sizeof( struct foo )`? _____

lower memory
fubar:

higher memory

3. For the following C expressions, give an equivalent expression that evaluates to the same value. In cases where there is more than one possible answer give the simplest expression with the least number of operators.

```
int a[5];        /* int[5] a;  -- Reduced-C syntax */
```

&a[3]   _____

*a   _____

a   _____

&*a   _____

*&a[4]   _____

a[1]   _____

If `a[0]` is allocated at memory location 4000, at what memory location is `a[4]`? _____

4. Use of typedefs in Reduced-C to define composite types

Using Reduced-C syntax, define an array of 9 pointers to bool named `fool` (rhymes with bool) such that

```
bool b = true;

fool[8] = &b;
b = *fool[8];
```

are valid expressions. This will take two lines of Reduced-C code.

What question would you like to see on the Midterm?