Login name _____ **Quiz 2** **Name** _____

CSE 131

Signature _____ **Spring 2009** **Student ID** _____

1. Check #5 Given the following Reduced-C definitions:

```
function : float foo1( float & a ) { int b; return b; }
function : float foo2( 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 (which is similar to C++ rules). Use the letters associated with the available errors in the box below.

```
x = foo1( 4.2 );        ____
x = foo1( y );          ____
x = foo1( x );          ____
x = foo1( foo2( x ) );  ____
x = foo1( x + y );      ____
x = foo2( x );          ____
x = foo2( 4.2 );        ____
x = foo2( foo1( x ) );  ____
x = foo2( &x );         ____
x = foo2( y );          ____
x = foo2( x + 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

2. Modifiable L-vals, Non-Modifiable L-vals, R-vals

Using the Reduced-C Spec (which closely follows the real C language standard), given the definitions below, indicate whether each expression evaluates to either a

A) Modifiable L-val B) Non-Modifiable L-val C) R-val

```
function : int * foo() { /* Function body not important. */ }
float[9] a;
float x;
const float y = 5.5;
float *p = &x;
```

____ foo()	____ 4.2	____ *foo()	____ (int)x	____ *(int *)p
____ p	____ a[2]	____ (int *)&x	____ *(int *)&x	____ (float *)foo()
____ x	____ *p	____ **&p	____ y	____ *foo() * y
____ &x	____ a	____ &*p	____ *p - y	____ *(float *)foo()
____ x++	____ ++x	____ a[2]++	____ &a[0]	____ ++*foo()

3. Type Inference (Mostly from Check #1). The types in Reduced-C variable definitions are often unnecessary in the sense that it may be possible to infer variables' types and detect type errors simply from their use. For each of the following program fragments, find a set of types that makes it legal, and write a Reduced-C definition for each variable. If there is more than one possible type, choose only one. If there is none, write "NONE". Assume all arrays are of size 9.

```
    a = 5.5;

    if ( c != b )
        c = (d % b) / a;
```

_____ a ;

_____ b ;

_____ c ;

_____ d ;

```
    if ( c && (b != a) )
        c = b;
```

_____ a ;

_____ b ;

_____ c ;

```
    a = (*b)[a];
```

_____ a ;

_____ /* b requires two lines of Reduced-C */

_____ b ; /* to properly define it */

---

4. Identify the following C constructs as either

A) Pure Declaration B) Definition

____ extern int x; ____ int foo(int x) { return x; }

____ struct fubar { int x; } s1; ____ float y;

____ struct fifi; ____ extern int * func1(int x, float y);