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

CSE 131

Signature _____ **Winter 2010** **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; }
function : float foo3( int a )     { int b; return b; }

int x;  /* global variables */
float 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.

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

```
y = foo1( y );          ____
y = foo1( &y );         ____
y = foo3( y );          ____
y = foo2( foo1( y ) );  ____
y = foo1( x + y );      ____
y = foo1( x );          ____
y = foo3( (int) y );    ____
y = foo1( foo2( y ) );  ____
y = foo2( &x );         ____
y = foo2( x );          ____
y = foo2( x + y );      ____
```

```
y = foo1( (float) x );          ____
y = foo3( x + y );              ____
y = foo1( 42.37 ):              ____
y = foo3( x + x );              ____
y = foo3( (int) foo1( y ) );    ____
```

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 : float * foo() { /* Function body not important. */ }
bool[9] a;
int x;
const bool y = true;
structdef S1 { int a; float b; bool c; };
S1 z;
S1 *p = &z;
```

```
____ foo()   ____ 42      ____ *foo()        ____ (float)x       ____ *(int *)p
____ p       ____ a[2]    ____ (float *)&x   ____ *(float *)&x   ____ (int *)foo()
____ y       ____ *a      ____ *&x           ____ *foo() * x     ____ *(int *)&p->b
____ &x      ____ a       ____ y != false    ____ x = z.a        ____ *(int *)foo()
____ x++     ____ ++x     ____ z.c           ____ p->b           ____ ++*foo()
```

3. Given the following Reduced-C statements, state which kind of STO (VarSTO, ConstSTO, ExprSTO, or ErrorSTO) is created and what is the resulting Type (bool, int, float, or null for an Error) stored in the STO and what value is stored in the STO if there is one (for example true, 42, 4.2, or none). Assume you are following our advice on which STOs to use in certain situations. A number in a later expression represents the appropriate previously parsed expression.

```
int a = 5;
const int b = 2;
const float c = 7.5;
const bool d = false;
float e;
e = (((b + c) * b - c / a) > a) != (d || true);
```

	Type of STO	Resulting Type	Value
#1: a	_____	_____	_____
#2: b	_____	_____	_____
#3: c	_____	_____	_____
#4: d	_____	_____	_____
#5: (b + c)	_____	_____	_____
#6: #5 * b	_____	_____	_____
#7: c / a	_____	_____	_____
#8: #6 - #7	_____	_____	_____
#9: #8 > a	_____	_____	_____
#10: d || true	_____	_____	_____
#11: #9 != #10	_____	_____	_____
#12: e = #11	_____	_____	_____

4. How did your group detect return statements at the top level in a function?

5. Specify what underlying type is declared with each typedef (not the typedef name).
For example, int *, int [7], etc.

```
typedef int T;          T:  _____________________
typedef T T2;           T2: _____________________ (not T)
typedef T2[5] T3;       T3: _____________________
typedef T3[2] T4;       T4: _____________________
typedef T2* T5;         T5: _____________________
```

6. Method overloading is resolved at _____ time.

Method overriding is resolved at _____ time.