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

CSE 131

Signature _____ **Winter 2014** **Student ID** _____

1. Phase 0 Scoping Fix. Fill in the blanks of the following Reduced-C program with correct types to test if your fix to the scoping bug present in the starterCode works correctly. If the scoping bug is fixed, this program should compile without error. If the bug is not fixed, this program should generate an assignment error at the line a = z;

```
_______  a;      // global a

function : int main() {

  _______  a;    // local a

  int z;

  a = z;         // If fixed, this line will not cause an error!
                 // If not fixed, this line will cause an error!
  return 0;
}
```

2. 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) Non-Modifiable L-val B) Modifiable L-val C) R-val

```
function : int * foo1() { /* Function body not important. */ }
function : int & foo2() { /* Function body not important. */ }
float[9] a;
float x;
const float y = 5.5;
float *p = &x;
```

____ foo2()	____ 4.2	____ *foo1()	____ x = y	____ *(int *)p
____ p	____ a[2]	____ (int *)&x	____ *(int *)&x	____ a[2]++
____ y	____ *p	____ **&p	____ ++x	____ foo1()
____ &a[0]	____ a	____ &*p	____ *p - y	____ foo2

3. Given the following variable definitions, determine whether each line of code will cause

1) No Error
2) Syntax Error
3) Semantic Error

Treat each line independently.

```
int a = 42;
bool b = true;
float c = 4.20;

b = c < a;    _____      c = c + c     _____

a = a + c;    _____      a = a ^ 7;    _____

a = a+++a;    _____      a = a+++++a;  _____

a = a++a;     _____      (c = a) = 5;  _____
```

4. Given the following Reduced-C code fragment:

```
int a;
float b;

function : float & foo( float & x, int y )
{ /* function body */ }
```

Using variables `a`, `b`, and the expression `(a + b)` as possible arguments to the function `foo()` and assuming there is no error with the assignment of the returned value:

Give an example function call to foo() that triggers an addressability error (and only this error).

```
someVar = foo( _____ , _____ );
```

Give an example function call to foo() that triggers an assignability error (and only this error).

```
someVar = foo( _____ , _____ );
```

Give an example function call to foo() that triggers an equivalence error (and only this error).

```
someVar = foo( _____ , _____ );
```

Assume the function body above contains `float z = 4.20; return z;` Although this will compile and run, what kind of programming error would this be considered?

Assume the arguments to foo() above are correct and the return statement in foo() is correct. Indicate which of the following statements are valid in our Reduced-C compiler and which will cause a compile error.

```
foo( /* Correct args */ ) = a;      ____
foo( /* Correct args */ ) = b;      ____
a = foo( /* Correct args */ );      ____
b = foo( /* Correct args */ );      ____
```

A) No Error in Reduced-C
B) Addressability Error
C) Assignability Error
D) Equivalence Error

5. Identify the following C constructs as either

A) Pure Declaration B) Definition

```
____ struct fofo;          ____ extern int * func1( int x, float y );
____ int x;                ____ int foo( int x ) { return x; }
____ extern float y;       ____ struct fubar { int x; } s1;
```

6. In class I handed out in a Turing Award Lecture paper related to Compilers. Who gave this Turing Award lecture? _____

A) Brian Kernighan
B) Dennis Ritchie
C) Ken Thompson
D) Al Aho
E) John Backus
F) Grace Hopper
G) John von Neumann
H) Donald Knuth
I) Fran Allen