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

CSE 131

Signature _____ **Winter 2009** **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. Modifiable L-vals, Non-Modifiable L-vals, R-vals

From the Reduced-C Spec (which follows closely the real C language standard), complete the following:

A) Addressable C) Modifiable
B) Not Addressable D) Not Modifiable

A Modifiable L-value is _____ and _____.

A Non-Modifiable L-value is _____ and _____.

An R-value is _____ and _____.

Given the definitions below, indicate whether each expression is either a

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

```
int x;
const int y = 5;
int[5] a;
int *p = &x;
```

```
____ (float *)p    ____ *(float *)p    ____ (float *)&x    ____ *(float *)&x

____ &x     ____ a     ____ y     ____ x + y     ____ a[2]     ____ 42

____ x      ____ *p    ____ *&*p  ____ &*p       ____ p
```

3. Type Inference. Consider the following Reduced-C program:

```
const bool a = 5 _Op1_ 7;
const bool b = true _Op3_ false;
const int c = 5 _Op2_ 7;
float x;
bool z;

function : int main()
{
  if ( a ) { return 1; }

  z = b;
  x = c;
  return 0;
}
```

For **_Op1_**, **_Op2_**, and **_Op3_**, list what operators are valid (i.e., cause no compile errors). The available operators are listed below. Some _Op#_ have more than one possible valid operator.

`==` `&&` `+` `>=`

_Op1_: _____

_Op2_: _____

_Op3_: _____

4. Semantic Checks/Errors:
Given the following Reduced-C code fragment:

```
int a;
bool b;

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

Using variables `a`, `b`, and the expression `(a + 2)` as possible arguments to the function `foo()`

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

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

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

In Reduced-C (which again follows closely the real C standard) all typedefs use _____ name equivalence. Struct operations (like =, ==, !=) use _____ name equivalence.