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

CSE 131

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

1. Regarding type equivalence and assignability, indicate whether each statement is

A) Always true
B) Sometimes true
C) Never true

_____ If type T1 is equivalent to type T2, then type T2 is assignable to type T1.

_____ If type T1 is not equivalent to type T2, then type T1 is not assignable to type T2.

_____ If type T1 is assignable to type T2, then type T1 is equivalent to type T2.

_____ If type T1 is not assignable to type T2, then type T2 may be equivalent to type T1.

2. Show the memory layout of the following C struct definition taking into consideration the **SPARC** data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate struct 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[3];
   double b;
   char   c;
   short  d[5];
};

struct foo fubar;
```

What is the `offsetof( struct foo, d[0] )`? _____

What is the `offsetof( struct foo, a[0] )`? _____

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

If `struct foo` had been defined as `union foo` instead, what would be the `sizeof( union foo )`? _____

lower memory

fubar:

higher memory

3. Given the array declaration

C	**Reduced-C**
`int a[7];`	`int[7] a;`

Mark with an **A** the memory locations (each box represents a byte in memory) where we would find

```
      a[5]
a:
```

```
low memory                                                    high memory
```

If `a[0]` is allocated at memory location 4000, at what memory location is `a[3]`? _____

4. Given the C following definitions

```
struct S1 { int a; };
struct S2 { int a; };

void foo1( float a ) { }
void foo2( S1 &a ) { }

typedef int T1;
typedef T1 T2;
typedef S1 T3;
typedef T3 T4;

S1 a;
S2 b;
T1 c;
T2 d;
T3 e;
T4 f;
int g;
```

indicate whether each of the following statements will cause a compiler error or not.

A) No Error
B) Error

`a = b;` _____	`foo1( c );` _____
`a = e;` _____	`foo1( d );` _____
`a = f;` _____	`foo2( a );` _____
`c = d;` _____	`foo2( b );` _____
`c = g;` _____	`foo2( e );` _____
	`foo2( f );` _____

Using the Rt-Lt Rule, give the C variable definition for a variable named foo that is a pointer to a function that takes a pointer to an int and returns a pointer to an array of 7 elements where each element is a pointer to an int.