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

CSE 131

Signature _____ **Winter 2010** **Student ID** _____

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 the following program fragment, 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".

```
d = 4.2;

if( a != b )
    a = (c & b) / d;

_____________________ a ;

_____________________ b ;

_____________________ c ;

_____________________ d ;
```

```
function : bool foo( float &x ) {...}

if( c = foo( b ) == a )
    d = &b;

_____________________ a ;

_____________________ b ;

_____________________ c ;

_____________________ d ;
```

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 {
   int     a;
   char    b;
   short   c[2];
   float   d;
   double *e;
   double  f;
   int     g;
};

struct foo fubar;
```

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

What is the `offsetof( struct foo, f )`? _____

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**
`float a[6];`	`float[6] a;`

Mark with an **A** the memory location(s) where we would find `a[3]`

(each box represents a byte in memory)

`a:`

`low memory` `high memory`

If `a[0]` is allocated at memory location 8000, what does `&a[5]` evaluate to? _____

4. Given the C following definitions (same rules in our Reduced-C compiler)

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

struct S1 a;
struct S2 b;
```

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

A) No Error
B) Error

_____ `a = b;`	_____ `a.a = b.a;`
_____ `a = (struct S1)b;`	_____ `(struct S2)a = b;`
_____ `a = *(struct S1 *)b;`	_____ `*(struct S2 *)a = b;`
_____ `a = *(struct S1 *)&b;`	_____ `*(struct S2 *)&a = b;`

5. Define an array of array of ints named `bar` in Reduced-C such that

```
bar[4][8]
```

is the last element in this data structure. You will need two lines of Reduced-C code to do this.

6. If there is a compile error in a Reduced-C source file, what should be the very last line of your compiler output?