Signature ___________________ **Quiz 3** **Name** ______________________

CSE 131

Login name _________________ **Winter 2014** **Student ID** __________________

1. 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;
   double  c;
   short   d[3];
   char    e;
   short   f;
   short   g;
   char    h;
};

struct foo fubar;
```

lower memory

fubar:

higher memory

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

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 )`? ________

2. Given the C array declaration

C

```
long long b[3];
```

Mark with an **B** the memory location(s) where we would find `b[2]`

(each box represents a byte in memory)

b:

low memory high memory

If `b[0]` is allocated at memory location 8000 (decimal), what value does `&b[1]` evaluate to? ____________

3. Given the following Reduced-C definitions:

```
function : float foo( float & a ) { int b; return b; }

float x;  /* global variables */
int 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. Use the letters associated with the available errors in the box below.

```
 x = foo( 4.2 );          ____
 x = foo( y );            ____
 x = foo( x );            ____
 x = foo( foo( x ) );     ____
 y = foo( x );            ____
 x = foo( x + y );        ____
&x = foo( x );            ____
 x = foo( &x );           ____
```

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
E) Left-hand operand is not assignable (not a mod L-val)
F) Value of right-hand-side type not assignable to left-hand-side type

4. Using Reduced-C syntax, define an array of array of ints named `foo` such that

```
foo[9][5]
```

is the last element in this data structure. This will take two lines of Reduced-C code.

5. Using only the following C variable declarations:

```
int a = 42;
int *aPtr = &a;
float b = 4.20;
float *bPtr = &b;
```

Using only `a` and `b` above, give an example assignment stmt using a converting type cast (underlying bit pattern changes).

_________________________________ = _________________________________ ;

Using only `aPtr` and `bPtr` above, give an example assignment stmt using a non-converting type cast (underlying bit pattern does not change).

_________________________________ = _________________________________ ;

6. Using the C language Rt-Lt Rule, declare `foo` to be a function that takes a pointer to a char and returns a pointer to an array of 37 elements where each element is a pointer to a struct bar.