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.

```c
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 )`? ________
3. Given the array declaration

\[ \text{C} \]
int a[7];

\[ \text{Reduced-C} \]
int[7] a;

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

\[
\begin{array}{cccccccc}
\text{a[5]} & \text{a[6]} & \text{a[7]} & \text{a[0]} & \text{a[1]} & \text{a[2]} & \text{a[3]} & \text{a[4]}
\end{array}
\]

low memory

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

4. Given the C following definitions

```c
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

\[
\begin{align*}
a &= b; & \text{foo1( c ); } & \text{_____} \\
a &= e; & \text{foo1( d ); } & \text{_____} \\
a &= f; & \text{foo2( a ); } & \text{_____} \\
c &= d; & \text{foo2( b ); } & \text{_____} \\
c &= g; & \text{foo2( e ); } & \text{_____} \\
& & \text{foo2( f ); } & \text{_____}
\end{align*}
\]

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.