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

CSE 131B

Signature _____ **Spring 2006** **Student ID** _____

1. Given the following C definitions

```
int array[5] = { 1, 2, 3, 4, 5 };
int *ptr = array;
```

what is the type of each expression below and is the result of the expression a modifiable l-val (mlv) (y or n)?

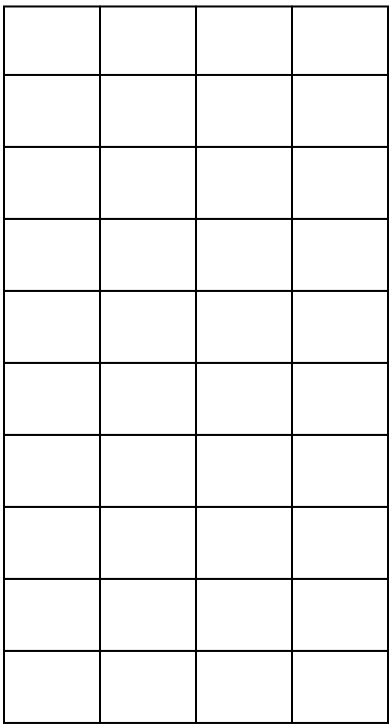
	Type	mlv?		Type	mlv?
`array`	_____	____	`++ptr`	_____	____
`ptr[2]`	_____	____	`&ptr[0]`	_____	____
`&ptr`	_____	____	`++*ptr`	_____	____
`*ptr++`	_____	____	`&*(array+2)`	_____	____

2. Show the memory layout of the following C struct/record definition taking into consideration the **SPARC** data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate struct/record 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 `p0`s, `p1`s, `p2`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;
   short  b[2];
   double c;
   char   d;
   int    e[3];
};

struct foo fubar;
```

fubar:



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

What is the `offsetof( struct foo, e[1] )`? _____

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

3. Give an example of a non-converting type cast (underlying bit pattern does not change).

4. For the following Oberon statements, indicate the correct error message using the list of given error messages below (if there is no error, select option A):

Possible Error Messages:
A - No error
B - Incompatible type to binary operator
C - Incompatible type to unary operator
D - Is not assignable (not a modifiable L-value)
E - BOOLEAN required for conditional test
F - Argument not assignable to value parameter
G - Argument not equivalent to REF parameter
H - Non-addressable argument passed to REF parameter

```
CONST t = TRUE;
TYPE foo = INTEGER;
TYPE bar = REAL;
TYPE baz = BOOLEAN;
VAR x : POINTER TO foo;
VAR y : bar;
VAR z : baz;
FUNCTION p(REF a : REAL; b : REAL) : foo;
  RETURN 0;
END p;

BEGIN
 y := ~z;              ______

 x&^^ := 99;           ______

 p(x^ / 1.0, y);       ______

 p(z, 9);              ______

END.
```

What question would you most like to see on the Midterm?