Signature _____ **Name _____**

Login Name _____ **Student ID _____**

Midterm
CSE 131B
Spring 2005

Page 1	**_____ (20 points)**
Page 2	**_____ (18 points)**
Page 3	**_____ (22 points)**
Page 4	**_____ (20 points)**
Page 5	**_____ (20 points)**
Subtotal	**_____(100 points)**
Page 6 Extra Credit	**_____ (5 points)**
Total	**_____**

1. Consider the following psuedocode:

```
x : string;             -- global x

procedure set_x ( n : string )
      x := n;

procedure print_x()
      output( x );      -- print the value of x

procedure one()
      x : string;
      set_x( "RED" );
      print_x();

procedure two()
      set_x( "GREEN" );
      print_x();

-- main starts here
set_x( "BLUE" );
one();
print_x();
two();
print_x();
```

What does the program output if the language uses <u>static</u> scoping? (4 points)

_____
_____
_____
_____

What does the program output if the language uses <u>dynamic</u> scoping? (4 points)

_____
_____
_____
_____

Give an example of a converting type cast/conversion (underlying bit pattern needs to be/is changed). (4 points)

Give an example of an implicit type coercion (type conversion without an explicit cast). (4 points)

Give an example of a non-converting type cast/conversion (underlying bit pattern does not change). (4 points)

2. Given the array declaration

C

```
int a[2][3];
```

Oberon-like

```
VAR a : ARRAY 2,3 OF INTEGER;
```

Mark with an **A** the memory location(s) where we would find

```
      a[1][1]                          a[1,1]
a:
```

```
low memory                                                    high memory
```

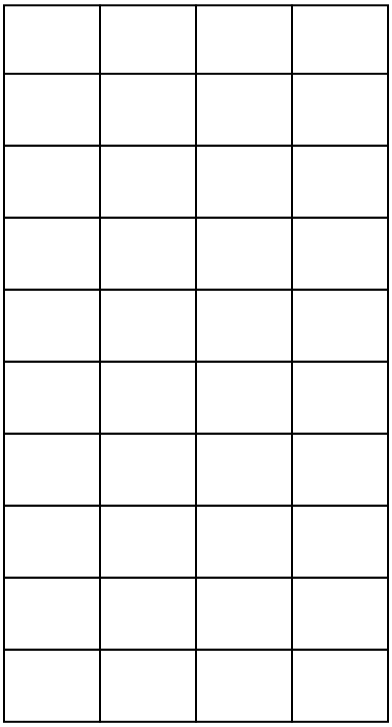
Each box represents a byte in memory. (4 points)

Show the SPARC memory layout of the following 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. (8 points)

```
struct foo {
   short  a;
   int    b;
   char   c[5];
   double d;
   float  e;
}

struct foo fubar;
```

```
                                                    low memory
fubar:
```



```
                                                    high memory
```

What is the `offsetof( struct foo, d )`? _____ (2 point)

What is the `sizeof( struct foo )`? _____ (2 point)

If `struct foo` had been defined as `union foo` instead, what would be the `sizeof( union foo )`? _____ (2 points)

3. Write a valid Oberon program to perform a simple test of the Project I POINTER type and NEW construct. Declare a global integer pointer named `ptr`, use NEW with this pointer appropriately, then assign the value 777 to the newly created integer. (11 points)

The types in Oberon variable declarations are often unnecessary in the sense that it is possible to infer variables' types and detect type errors simply from their use. However, adding type declarations can change the meanings of certain programs, or make them illegal. Consider the following program fragment: (11 points)

```
IF x1 = x2 THEN
    x3 := x4 DIV 7 + x1;
END;
```

Assume all variables have type INTEGER, REAL, or BOOLEAN.

Specify a valid type for each variable so that the code fragment above is legal (no error).

`x1`

`x2`

`x3`

`x4`

Now given the following declaration:

```
VAR x1: REAL;
```

State the (possibly new) type of each the following variables so that the code fragment above is legal (no error).

`x2`

`x3`

`x4`

What error, if any, would result if `x2` were defined to be of type BOOLEAN instead of what you just specified?

What error, if any, would result if `x4` were defined to be of type REAL instead of what you just specified?

4. Consider the following Oberon code:

```
VAR i : INTEGER;
VAR b : BOOLEAN;
VAR r : REAL;
VAR a : ARRAY 10 OF INTEGER;
```

Example 1:

```
BEGIN
   a[10] := a[5 + r];
END.
```

Describe all the errors (if any) in the above statement in general terms. State "None" if there are no errors.

Example 2:

```
BEGIN
   IF r < 5.5 THEN r := i; EXIT; END;
END.
```

Describe all the errors (if any) in the above statement in general terms. State "None" if there are no errors.

With regard to our Nano-Oberon specification,

Can a variable of type record be passed by value (non-VAR) to a procedure? _____

Can a variable of type record be passed by reference (VAR) to a procedure? _____

Can a variable of type array be passed by value (non-VAR) to a procedure? _____

Can a variable of type array be passed by reference (VAR) to a procedure? _____

Can a variable of type boolean be passed by value (non-VAR) to a procedure? _____

Can a variable of type boolean be passed by reference (VAR) to a procedure? _____

With regard to this C/C++ array: `double arr[5];`

What is the `sizeof( arr )`? _____

If `&arr[0]` is 8000, what is `&arr[3]`? _____

What type is `&arr[2]`? _____

What type is `*&arr[2]`? _____

5. Describe what each part of the C compilation → program execution process does/is responsible for:

Assembler ...

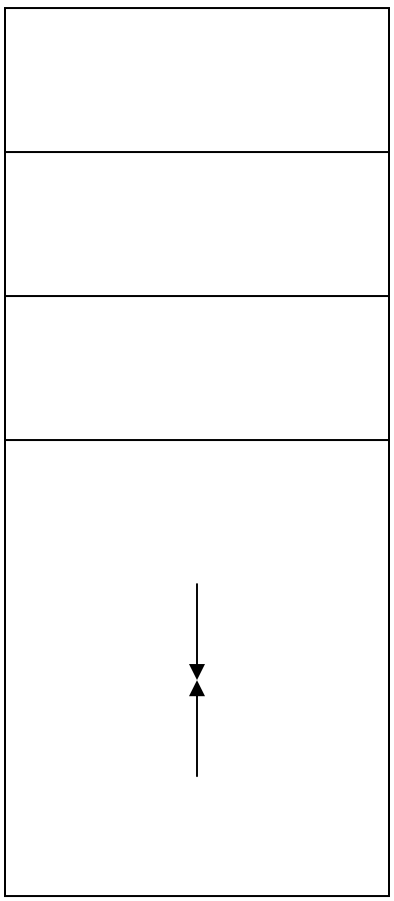
Loader ...

C Preprocessor ...

Linkage Editor ...

C Compiler ...

Fill in the names of the 5 areas of the C Runtime Environment as laid out by most Unix operating systems (and Solaris on SPARC architecture in particular) as discussed in class. Then state what parts of a C program are in each area. (10 points)



Extra Credit (5 points)

What is the value of each of the following expressions?

```
char a[] = "This 1 Blows ME Away!";
char *ptr = a;

++*(a + 18)          _____________

5[ptr] + 2           _____________

++*a                 _____________

*(&ptr[14]-7)        _____________

ptr[strlen(ptr)-7]   _____________
```