Signature _____ **Name** _____

Login Name _____ **Student ID** _____

Midterm
CSE 131B
Spring 2004

Page 1	_____ **(24 points)**
Page 2	_____ **(17 points)**
Page 3	_____ **(27 points)**
Page 4	_____ **(21 points)**
Page 5	_____ **(16 points)**
Subtotal	_____**(105 points)**
Page 6 Extra Credit	_____ **(5 points)**
Total	_____

1. Fill in the blanks/matching (1 point each); short answer (2 points each).

_____ analysis deals with verifying correct structure of a program.

_____ analysis deals with verifying correct meaning of a program.

What is the difference between an interpreter and a translator?

An interpreter ...

A translator ...

A) Never	D) Boot time	G) Once
B) Load time	E) Run time	H) Each time it is accessed
C) Link time	F) Each time the function in which it is defined is called	

For the following, use the correct LETTER from the choices above for a typical C program:

In C, when is the memory location that has been allocated for an external static variable initialized? ____

In C, when is the memory location that has been allocated for an initialized local variable initialized? ____

In C, how many times are global variables initialized? ____

In C, how many times are external static variables initialized? ____

In C, when is the memory location that has been allocated for an internal static variable initialized? ____

In C, how many times are internal static variables initialized? ____

In C, when is the memory location that has been allocated for a global variable initialized? ____

In C, how many times are initialized local variables initialized? ____

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

Assembler ...

C Preprocessor ...

Loader ...

C Compiler ...

Linkage Editor ...

2. Given the array declaration

C	**Oberon-like**
`int a[3][2];`	`VAR a : ARRAY 3,2 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
```

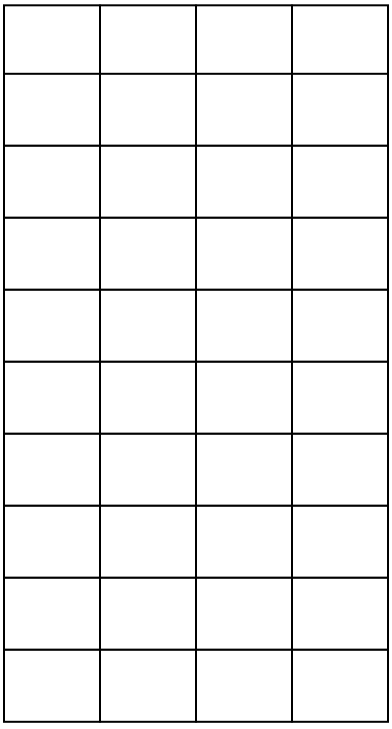
Each box represents a byte in memory.

2. Show the 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 are padded so the total size is evenly divisible by the most strict alignment requirement of its members. (9 points)

```
struct foo {
   char   a;
   int    b;
   double c;
   short  d;
   char   e[4];
   float  f;
}

struct foo fubar;
```

```
                                                    low memory
fubar:
```



```
                                                    high memory
```

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

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

3. Give an example of an assignability error.

Give an example of a type equivalence error.

Give an example of an addressability error.

Regarding type checking, reference (VAR) parameters require the actual arguments to be _____ and _____ to the formal parameter type while value parameters require the actual arguments to be _____ to the formal parameter type. (2 points each)

Give an example of an implicit type coercion (type conversion without an explicit cast). (3 points each)

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

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

Give an example of a type inference rule the compiler will perform.

4. Consider the following Oberon code and your Project I Semantic Type Checking:

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

Example 1:

```
a[10] := a[3,r];
```

How many errors (if any) would this statement generate? Describe the error(s) / error message(s) in general terms.

Example 2:

```
r := i + r - i;
```

How many errors (if any) would this statement generate? Describe the error(s) / error message(s) in general terms.

Example 3:

```
VAR x, y : INTEGER;
VAR z : REAL;
VAR rPTR : POINTER TO REAL;

PROCEDURE P ( a : INTEGER; VAR b : INTEGER );
BEGIN
      ...
END P;

BEGIN
   NEW( z );
   P( rPTR^, x + z + y );
END.
```

How many errors (if any) would this code generate? Describe the error(s) / error message(s) in general terms.

5. In a strongly-typed language like Oberon, we can perform array range checking and be confident that an array access expression like

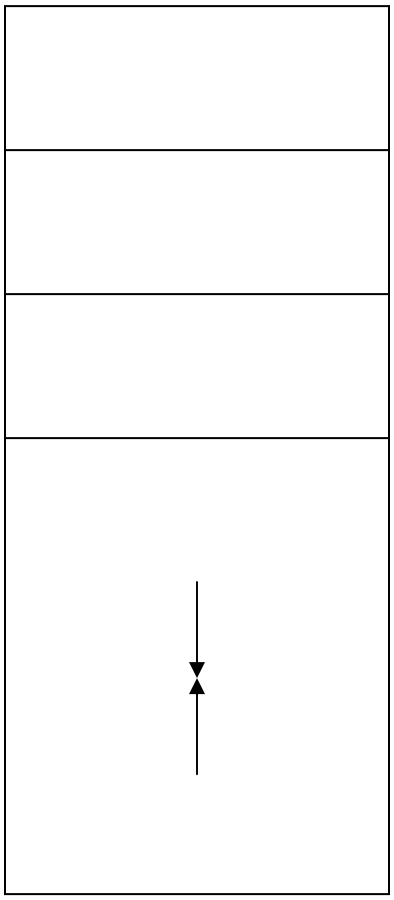
```
a[-5]
```

is an error. In a language like C that allows arrays and pointers to be treated similarly, this may not be an error.

In what context could the C compiler not be confident this is an error and thus not report an error? Give an example. (3 pts)

In what context could the C compiler be somewhat confident the above expression is an array out-of-bounds error? Give an example. (3 pts)

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 Blows Me Away!";
char *ptr = a;

*&a[8]                      _____________

13["End this, please!"]     _____________

*(&ptr[5]-4)                _____________

*(a + 7)                    _____________

"I loved Project I!"[3]     _____________

ptr[18]                     _____________
```