Signature ___________________  Name ______________________
Login Name __________________  Student ID ________________

Midterm  
CSE 131B  
Winter 2006  

<table>
<thead>
<tr>
<th>Page</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 1</td>
<td>26</td>
</tr>
<tr>
<td>Page 2</td>
<td>17</td>
</tr>
<tr>
<td>Page 3</td>
<td>20</td>
</tr>
<tr>
<td>Page 4</td>
<td>19</td>
</tr>
<tr>
<td>Page 5</td>
<td>18</td>
</tr>
</tbody>
</table>

Subtotal (100 points)  

Page 6 (5 points)  

Total
1. Give the order of the phases of compilation in a typical compiler as discussed in class

A – Machine-specific code improvement (optional)  
B – Parser (Semantic analysis)  
C – Machine-independent code improvement (optional)  
D – Source language file (for example, C)  
E – Target language file (for ex., assembly)  
F – Scanner (lexical analysis)  
G – Parser (syntax analysis)  
H – Code generation (for ex., assembly)  
I – Intermediate Representation


Give the order of the typical C/C++ compilation stages and on to actual execution as discussed in class

A – Program Execution  
B – as (assembler)  
C – Source file (.c/.cpp)  
D – cpp (C preprocessor)  
E – Segmentation Fault (Core Dump)  
F – ccomp (C compiler)  
G – ld (Linkage Editor)  
H – exe/a.out (executable image)  
I – loader


Given the following ANSI/ISO C/C++ variable definitions, which line(s) would cause semantic compiler errors?

A. Compiler error  
B. No compiler error

```
it i;
int * iPtr = &i;
int ** pPtr = &iPtr;

*pPtr++;  ______
++(&i);  ______
++*pPtr++;  ______
++(*pPtr)++;  ______
******+pPtr++;  ______
******+iPtr;  ______
******+pPtr;  ______
******+pPtr++;  ______
```
2. Given the array declaration

C
int a[3][3];

Oberon-like
VAR a : ARRAY 3,3 OF INTEGER;

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

\[
\begin{array}{c}
a[2][1] \\
a[2,1]
\end{array}
\]

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. (7 points)

```
struct foo {
    char   a;
    short  b[2];
    double c;
    char   d[10];
    char   e[3];
    int  *f;
    char   g;
}
```

```
struct foo fubar;
```

What is the `offsetof( struct foo, c )`? _______ (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. 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): (2 pts each)

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 VAR parameter
H - Non-addressable argument passed to VAR parameter

```
CONST t = TRUE;
TYPE foo = INTEGER;
TYPE bar = REAL;
TYPE baz = BOOLEAN;
VAR x : foo;
VAR y : bar;
VAR z : baz;
PROCEDURE p(a : REAL; VAR b : REAL);
(* do nothing *)
END p;

BEGIN
  y := 99;         ______
  z := x # y;         ______
  z := ~x;         ______
  t := z;         ______
  p(x, x);         ______
  p(9, 9.0);         ______
  p(x, y);         ______
  p(x DIV 1, y);         ______
  p(z, y);         ______
  IF (z & ~t) THEN END; ______
END.
```
4. Consider the following C-like code:

```c
int x = 0;

int f()
{
    print(x);
    return x;
}

int g()
{
    int x = 1;
    print(x);
    return f();
}

int main()
{
    print(g());
}
```

What does the program output if the language uses static scoping? (3 points)  
What does the program output if the language uses dynamic scoping? (3 points)

_____        _____
_____        _____

Consider the following record/struct definitions:

A
struct foo {
    int a;
    double b;
    struct foo c;
    short d[4];
};

B
struct foo {
    int a;
    double b;
    struct foo c[2];
    short d[4];
};

C
struct foo {
    int a;
    double b;
    struct foo *c;
    short d[4];
};

Which of the above record/struct definitions is/are semantically correct and why? (4 points)

Using the Right-Left rule write the definition of a variable named CSE that is a pointer to an array of 8 pointers to functions that take a pointer to a float as the single parameter and returns a pointer to a double. (9 points)
5. Given the following array definition

```c
/* C */
float x[3][5];

(* Oberon *)
VAR x : ARRAY 3,5 OF REAL;
```

write the assembly level address calculation expression **taking into account scalar arithmetic** to access

\[
\text{x[a][b]} \quad \text{x[a,b]}
\]

\[
(( \text{x} + \text{_______________________________}) + \text{_______________________________})
\]

The result is the address of where we can find this array element. (8 points)

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)

Explain what is wrong with the following CUP rule/action. How would you fix this problem?

```
ExprList ::= Expr:_1 {:
    Vector v = new Vector();
    v.addElement(_1);
    RESULT = v;
    :}
  | ExprList T_COMMA Expr:_2 {:
    v.addElement(_2);
    RESULT = v;
    :}

;