Signature _____ **Name** _____

Login Name _____ **Student ID** _____

Midterm
CSE 131B
Spring 2007

Page 1	_____ **(21 points)**
Page 2	_____ **(18 points)**
Page 3	_____ **(21 points)**
Page 4	_____ **(18 points)**
Page 5	_____ **(22 points)**
Subtotal	_____**(100 points)**
Page 6 Extra Credit	_____ **(5 points)**
Total	_____

1. Assume the following definitions are correct: (2 pts each)

```
ALIAS rec1 = RECORD
       ptr : POINTER TO FLOAT;
     END;

ALIAS rec2 = RECORD
       ptr : POINTER TO rec1;
     END;

VAR ptr : POINTER TO rec2;
VAR a, b : FLOAT;
```

a) What type is `ptr^.ptr^` ?

b) What type is `ptr^.ptr^.ptr` ?

c) What type is `ptr^.ptr` ?

d) What type is `ptr^.ptr^.ptr^` ?

For the following, assume each statement is independent of each other. You cannot use the result of a previous statement in the answer for a subsequent question. You can only use the above vars and types. You cannot use NIL. You can assume loose name equivalence and RECORD assignment is valid throughout.

e) Write the Oberon statement to assign variable **`b`** the value of the float pointed to by `ptr` in `rec1`.

```
b :=
```

f) Write a valid statement with variable **`c`** (defined below) on the lhs (left hand side) of the assignment statement. You cannot use `c` or NIL.

```
VAR c : rec2;

c :=
```

g) Using the Right-Left rule write the C definition of a variable named screwball that is an array of 9 elements where each array element is of type pointer to a function which takes two arguments, a pointer to a pointer to a char and a float, and returns a pointer to an array of 6 elements where each array element is of type pointer to struct student. (9 pts)

2. The types in Oberon variable definitions are often unnecessary in the sense that it is possible to infer variables' types and detect type errors simply from their use. For each of the following program fragments, find a set of types that makes it legal, and write an Oberon definition for each variable. If there is more than one possible type, choose only one. If there is none, write "NONE". Assume all arrays are of size 4. (2 pts each)

```
    b := a[b^];

VAR a : _____________________________________________ ;

VAR b : _____________________________________________ ;
```

```
    IF a # c THEN
        c := d / (b MOD a);
    END

VAR a : _____________________________________________ ;

VAR b : _____________________________________________ ;

VAR c : _____________________________________________ ;

VAR d : _____________________________________________ ;
```

```
    IF (a # b) OR c THEN
        c := b;
    END

VAR a : _____________________________________________ ;

VAR b : _____________________________________________ ;

VAR c : _____________________________________________ ;
```

---

3. Check #19a and #19b: How did your group implement the management of and semantic checks associated with function overloading? Talk about (9 pts)
a) what you inserted into the symbol table

b) how you implemented the checking for illegal overloaded function definitions

c) how you implemented the checking for illegal overloaded function calls.

Given the following array definition

```
   /* C */
float x[25][33];
```

write the assembly level address calculation expression <u>taking into account scalar arithmetic</u> to access

```
x[i][j]
```

((x + _____) + _____)

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

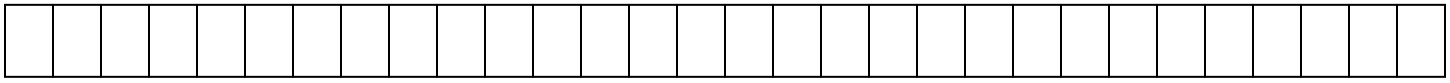
Given the array declaration

```
      C
short a[3][5];
```

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

```
a[1][1]
```

a:



low memory high memory

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

4. Briefly discuss 3 different ways/mechanisms discussed in class to subvert type checking in C. Multiple examples of the same general mechanism just counts as 1 way. (2 pts each on this page)

1)

2)

3)

What mechanisms/features of Java help to prevent the following which can/do regularly occur in C and are difficult for the compiler to detect?

a) dangling pointers/references into the Heap

b) dangling pointers/references into the Stack

c) memory leaks

What part(s) of a compiler make up the front end?

What part(s) of a compiler make up the back end?

Which class from Project I should be the only class that contains a member "m_value" to hold an actual value?

5. Write a valid Oberon program (according to this quarter's specs) to perform a simple test of the Project I Phase 0 change you made to the access() method in SymbolTable.java. What value would you expect the program to output before and after you made this Phase 0 fix? (12 pts)

Expected output before the fix _____

Expected output after the fix _____

Using the rules from Project I, classify the following assignment statements into one of the following:
A - No error
B - Non-modifiable L-value
C - Not structurally equivalent
D - Not strict name equivalent

```
ALIAS foo = RECORD x : INTEGER; END;
ALIAS bar = RECORD x : INTEGER; END;
ALIAS baz = foo;
ALIAS fubar = bar;
ALIAS fcuk = fubar;
VAR a,b : RECORD x : INTEGER; END;
VAR c : foo;
VAR d : bar;
VAR e : baz;
VAR f : fubar;
VAR g : fcuk;

BEGIN
   a := b;        _____

   a := c;        _____

   c := d;        _____

   e := c;        _____

   f := g;        _____

   RETURN 0;
END.
```

Extra Credit (5 points)

What gets printed by the following C program?

```
#include <stdio.h>

int
main()
{
  char a[] = "Me? I want to go";
  char b[] = "to Porter's Pub";
  char c[] = "and don't you, too?";
  char *ptr = b;

  printf( "%c\n", *(ptr = ptr + 3) );          _______

  printf( "%c\n", toupper( *c + 1 ) );         _______

  printf( "%c\n", b[strlen(a) - 2] );          _______

  printf( "%c\n", *(a + 7) );                  _______

  printf( "%c\n", ptr[5] );                    _______

  printf( "%c\n", ptr[5] + 1 );                _______
  return 0;
}
```

Scratch Paper