Login name ________________ **Name** ______________________

Signature __________________ **Student ID** __________________

Final
CSE 131B
Spring 2005

Page 1	__________	**(27 points)**
Page 2	__________	**(24 points)**
Page 3	__________	**(32 points)**
Page 4	__________	**(24 points)**
Page 5	__________	**(32 points)**
Page 6	__________	**(26 points)**
Page 7	__________	**(31 points)**
Page 8	__________	**(20 points)**
Subtotal	__________	**(216 points)**
Page 9 Extra Credit	__________	**(11 points)**
Total	__________	

1 Consider the following C++ function:

```
int fubar( int a )
{
  static int y = a;

  // Other possible code not relevant to this question

  return y;
}
```

Generate unoptimized SPARC assembly code for this function taking into consideration the semantics involved with the C++ language allowing static variables to be initialized at run time as discussed in class. (24 points)

Regarding the C++ function above: How does the compiler differentiate between the static variable y in function fubar() and a global variable named y or another static variable named y defined elsewhere in separate source files that may be part of the overall resulting program and keep their scope/visibility semantically correct? Be specific. Be sure to incorporate this into your code generation above. (3 points)

2. Given the following Oberon program and expected output, determine whether each parameter is pass-by-reference or pass-by-value. Fill in the blanks with "VAR" if pass-by-reference, leave it blank if pass-by-value.

```
VAR x : INTEGER;
VAR y : INTEGER;

PROCEDURE foo1 ( ____ a : INTEGER);
BEGIN
    a = y + 30;
END foo1;

PROCEDURE foo2 ( ____ a : INTEGER, ____ b : INTEGER);
BEGIN
    a = 2 * y;
    b = 3 * x;
END foo2;

PROCEDURE foo3 ( ____ a : INTEGER, ____ b : INTEGER);
BEGIN
    a = x + y;
    b = x - y;
END foo3;

BEGIN
    x = 10;
    y = 10;

    foo1(y);
    foo2(x, y);
    foo3(x, y);

    OUTPUT x, " ", y;     (* should output 80 60 *)
END.
```

Now in order to get full credit for the above and discourage Jeff Spicoli-like random guesses, what is the output if each VAR parameter was changed to non-VAR and each non-VAR parameter was changed to VAR?

You may use the space below as a scratch pad area to help figure out the correct parameter passing modes.

3. In your Project 2, how did you (and your partner if you had a partner) implement global variables? There were several possible implementation options discussed. Be specific how your project implemented them!
(10 points)

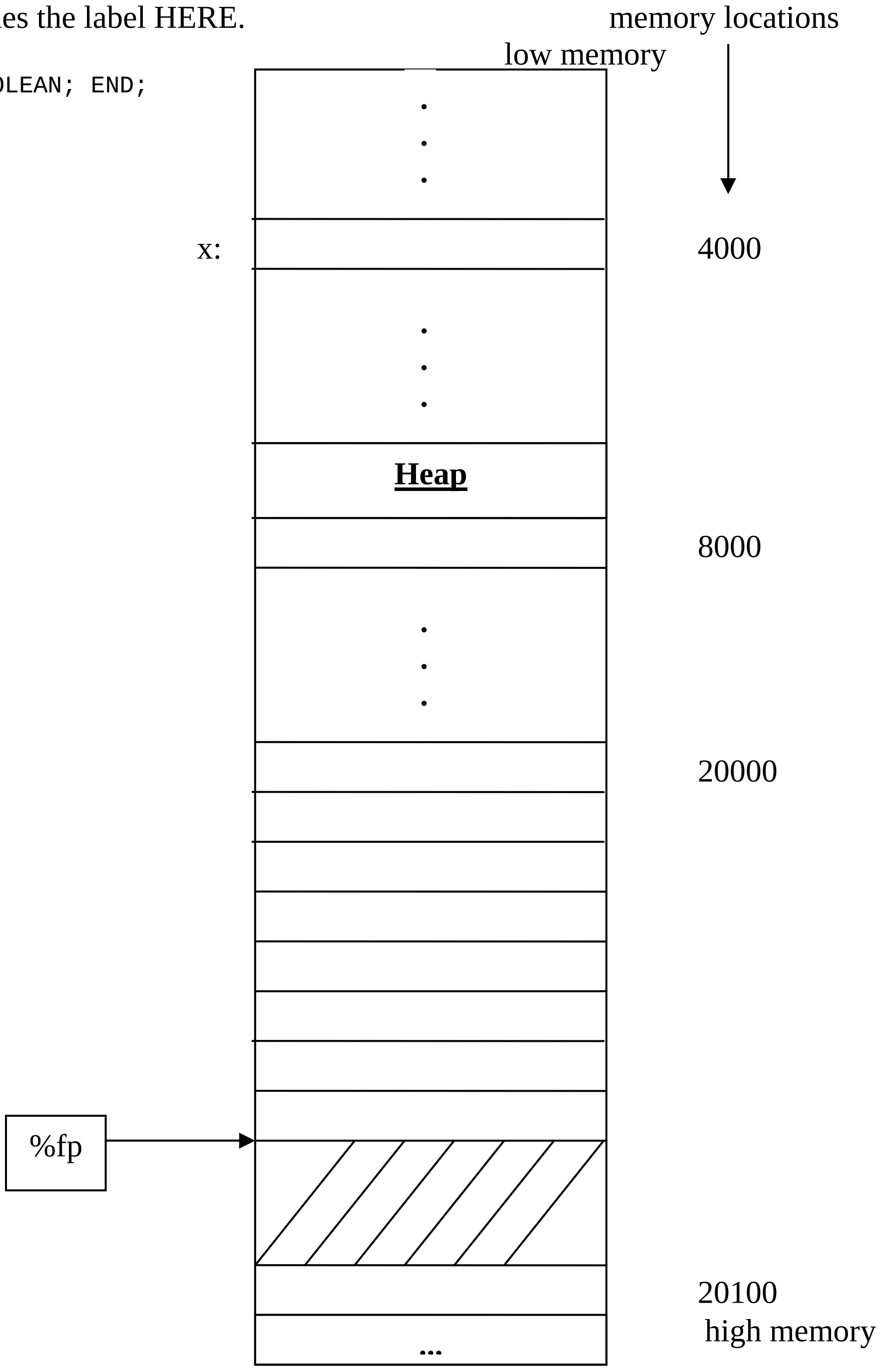
Given the following Oberon program, fill in the values of the global and local variables and parameter in the run time environment when the program reaches the label HERE.

```
TYPE t = RECORD a: INTEGER; b: BOOLEAN; END;
VAR x : INTEGER;

PROCEDURE f(VAR i: INTEGER);
    VAR j: INTEGER;
    VAR r2: POINTER TO INTEGER;
    VAR a2: ARRAY 3 OF t;
BEGIN
    NEW(r2);
    r2^ := 95;
    j := 17;
    a2[0].a := 5;
    a2[1].a := 13;
    a2[2].b := FALSE;
    a2[0].b := TRUE;
    a2[2].a := -28;
    a2[1].b := TRUE;

    (* HERE *)
END;

BEGIN
    f(x);
END.
```



4. Identify <u>where</u> each of the following program parts live in the Java runtime environment as discussed in class. (12 points)

```
public class Foo {
  private static Foo a;                         a           _________________

  private int b;                                b           _________________

  public Foo() {                                Foo()       _________________

    a = this;                                   this        _________________
    ++b;
  }
                                                main()      _________________

  public static void main( String[] args ) {    args        _________________

    Foo c = new Foo();                          c           _________________

    int d;                                      d           _________________

    c = new Foo();              where c is pointing         _________________
    c.method( d );
  }
                                                method()    _________________

  private void method( int e ) {                e           _________________

    int f;                                      f           _________________
    f = e;
  }
}
```

Using the Right-Left rule (which follows the operator precedence rules) write the definition of a variable named foo that is a pointer to a pointer to a function that takes a pointer to a struct Fubar as the single parameter and returns a pointer to a 4x7 2-D array where each element is a pointer to an array of 8 doubles. (6 points)

What is an advantage of storing multi-dimensional arrays as contiguous memory locations as opposed to arrays of arrays in C/C++? (3 points)

What is an advantage of storing multi-dimensional arrays as arrays of arrays as opposed to contiguous memory locations in C/C++? (3 points)

5. Given the following C program, order the printf() line numbers so that the values that are printed when run on a Sun SPARC Unix system are displayed from smallest value to largest value (note the address of the different program parts are printed, not their value). Also specify the corresponding C Run Time area where each program part will be allocated. (22 points)

```
void foo( int *, int );    /* Function Prototype */

int a = 420;

int main( int argc, char *argv[] )
{

    int b = 420;

    foo( &argc, b );

/*  1 */ (void) printf( "a --> %p\n", &a );
/*  2 */ (void) printf( "b --> %p\n", &b );
/*  3 */ (void) printf( "argc --> %p\n", &argc );
/*  4 */ (void) printf( "argv --> %p\n", &argv );
/*  5 */ (void) printf( "malloc --> %p\n", malloc(75) );
/*  6 */ (void) printf( "foo --> %p\n", foo );
}

void foo( int *c, int d ) {

    int e = 404;
    static int f;
    int g;

/*  7 */ (void) printf( "c --> %p\n", &c );
/*  8 */ (void) printf( "d --> %p\n", &d );
/*  9 */ (void) printf( "e --> %p\n", &e );
/* 10 */ (void) printf( "f --> %p\n", &f );
/* 11 */ (void) printf( "g --> %p\n", &g );
}
```

Line that prints the smallest value (low memory)
____
____
____
____
____
____
____
____
____
____
____
Line that prints the largest value (high memory)

	Run Time Area
a	___________
b	___________
c	___________
d	___________
e	___________
f	___________
g	___________
foo	___________
argv	___________
malloc	___________
argc	___________

List the two main tasks/functions of the linkage editor as discussed in class? (6 points)

1)

2)

How does the this reference (in Java) or the this pointer (in C++) become available/accessible in a method? (4 points)

6. Given the following Oberon program, emit the **unoptimized** SPARC assembly language code that should be generated for the two procedures foo() and foo1(). Assume no optimizations – treat each instruction separately without any knowledge of any previously computed/loaded/stored values that may still be in a register from a previous instruction. Draw a line between each group of assembly language instructions that represent the emitted code generated for each instruction and label them with the instruction number. All local variables must be allocated and accessed on the Stack (do not map them directly into a local register). (26 points)

```
PROCEDURE foo( a : INTEGER; VAR b : INTEGER ) : INTEGER;
  VAR i : ARRAY 2 OF INTEGER;        (**** Local Stack variables – Do not initialize to zero ****)
BEGIN
  a := i[1] * 5;        (* 1 *)
  b := a + 10;          (* 2 *)
  i[0] := b;            (* 3 *)
  RETURN b;             (* 4 *)
END foo1;

PROCEDURE foo1( VAR a : INTEGER; VAR b : INTEGER );
  VAR i, j : INTEGER;        (**** Local Stack variables – Do not initialize to zero ****)
BEGIN
  j := foo( i, b );    (* 5 *)
  a := foo( a, j );    (* 6 *)  (* Don't worry about any local variables not initialized *)
END foo;

BEGIN
  (* ... *)
END.
```

```
        .global foo, foo1, main

        .section ".text"
foo:                                               foo1:
```

7. 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 are padded so the total size is evenly divisible by the most strict alignment requirement of its members. (11 points)

```
struct foo {
   short  a[3];
   int    b;
   double c;
   char   d;
   short  e;
}

struct foo fubar;
```

`fubar:` `low memory`

Generate SPARC assembly code for the C function below. (20 points)

C

```
void foo( int i )
{
  while ( i < 20 )
  {
    printf( "%d\n", i );
    ++i;
  }
}
```

SPARC

8. Consider a typical if-then-else statement that will be executed many times such as being embedded in a loop. Is it more efficient to structure the condition so the statements you know will most likely to be executed are in the if/true block or the else/false block? <u>Why?</u> (8 points)

```
if ( condition )
{
    if/true block
}
else
{
    else/false block
}
```

Perform loop improvements on the following C code fragment so the loop should execute more efficiently even without any compiler optimization. (10 points)

```
int i;
char str[] = "e_superstore = color";

    /* other code */

for ( i = 1; (i – 1) < strlen( str ); i = i * 2 )
{
    str[i-1] = toupper( str[i-1] );
}
```

<u>Code improved version here</u>

```
int i;
char str[] = "e_superstore = color";

/* Modifications from this point down */
```

Tell me something you learned in this class that is extremely valuable and that you think you will be able to use for the rest of your programming/computer science career. (2 points if serious; you can add non-serious comments also)

9. Extra Credit (11 points)

What gets printed by the following C program?

```
#include <stdio.h>

int
main()
{
  char a[] = "End this, Please!";
  char *ptr = &a[6];

  printf( "%c\n", *ptr++ );                _____

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

  printf( "%c\n", ptr[2] );                _____

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

  printf( "%c\n", (*(a+7))-3 );            _____
  ptr = &a[8];
  printf( "%c\n", *--ptr );                _____

  return 0;
}
```

Given the following ANSI/ISO C variable definitions, identify which expressions will produce a static semantic compiler error. Hint: Think modifiable l-value.

A) No compiler error
B) Compiler error

```
      int i = 5;
      float f = 1.5;
      int *iPtr = &i;
      float *fPtr = &f;

      fPtr = &(i + f);          ______

      ++( (float *) iPtr );     ______

      i = *(int *)fPtr;         ______

      iPtr = *(int **)&fPtr;    ______
```

Crossword Puzzle (next page)

Scratch Paper

Scratch Paper