1. What gets printed if the following function is invoked as `recurse( 3, 2 )` ?
(Draw stack frames on the side.)

```
int
recurse( int a, int b ) {
  int local = a - b;
  int result;

  printf( "%d\n", local );

  if ( a < 8 )
    result = local + recurse( a + 2, b );
  else
    result = local;

  printf( "%d\n", result );

  return result;
}
```

Put answers here

2. Given the following Oberon code fragment:

Oberon

```
PROCEDURE foo( i : INTEGER; VAR j : INTEGER ) : INTEGER;
BEGIN
  RETURN i + j;
END foo;
```

Write the equivalent translated SPARC Assembly language statements emitted for Procedure foo().

3. Given the following code for foo(), <u>write an equivalent more highly optimized version in SPARC assembly</u>.
Assume: x is mapped to local register %l1
                    y is mapped to local register %l2
                    z is mapped to local register %l3
                    a is a global variable allocated in the Data segment and NOT mapped to a register

Our Nano-Oberon

```
VAR a : INTEGER;

PROCEDURE foo( i : INTEGER );
  VAR x, y, z : INTEGER;

BEGIN
  x := x + 0;
  y := x - 5;
  z := y + i;



  a := x - 5;
  a := a * y;

END foo;



BEGIN
  foo( -7 );
END.
```

SPARC Assembly

```
        .global  main, foo

        .section ".data"
        .align 4
a:      .word 0

        .section ".text"
foo:                            ! mapping local vars
        save    %sp, -96, %sp   ! to local regs
        clr     %l1
        clr     %l2
        clr     %l3
        add     %l1, %g0, %l1   ! r1 = r1 + 0
        sub     %l1, 5, %l2     ! r2 = r1 - 5
        add     %l2, %i0, %l3   ! r3 = r2 + param1
        sub     %l1, 5, %l0     ! tmp = r1 - 5
        set     a, %l4
        st      %l0, [%l4]      ! a = tmp
        set     a, %l4
        ld      [%l4], %o0      ! out param1 = a
        mov     %l2, %o1        ! out param2 = r2
        call    .mul            ! a * r2
        nop
        st      %o0, [%l4]      ! a = a * r2

        ret
        restore

main:
        /* Code for main() not important */
```

Rewrite only code that is within the rectangle.

There are many optimization opportunities. Some are worth more than others.

What question would you most like to see on the Final?