1) Which part of the entire compilation sequence clear through to program execution is responsible for:

- resolving undefined external references with defined global references across modules
- translating assembly source code into object target code
- ensuring the bss segment is set up and zero-filled
- getting the executable image from disk into memory
- translating C source code into assembly target code
- creating an executable image from multiple object files

2) Specify the scope/visibility of each of the following:

- global variables ________
- local variables ________
- external static variables ________
- internal static variables ________
- formal parameters ________
- global functions ________
- static functions ________

3) Order the following storage hierarchy elements/types from fastest to slowest

- ________ (Fastest)
- ________
- ________
- ________
- ________ (Slowest)

4) What gets printed if the following function is invoked as `recurse( 3, 6 )`? Hint: Draw stack frames.

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

    if ( local <= 12 )
        result = local + recurse( a + 1, b );
    else
        result = local;

    printf( "%d\n", result );
    return result;
}
```

Put answers here

(over)
5) Given the following program, reorder the output so that the address values that are printed are sorted from smallest to largest if compiled and run on a Sun SPARC architecture. These lines print out the hex address of the different parts of the program (not the values assigned) with the printf() format specifier %p (pointer). Basically, where do the different parts of a C program live in the run time environment?

```c
#include <stdio.h>
#include <stdlib.h>

int a = 42;

void
foo( int b )
{
    int c;
    /* 1 */ (void) printf( "1:a --> %p\n", &a );
    /* 2 */ (void) printf( "2:b --> %p\n", &b );
    /* 3 */ (void) printf( "3:c --> %p\n", &c );
}

int
main( int argc, char *argv[] )
{
    int d = 420;
    int e;
    static int f;
    foo(d);
    /* 4 */ (void) printf( "4:foo() --> %p\n", foo );
    /* 5 */ (void) printf( "5:argc --> %p\n", &argc );
    /* 6 */ (void) printf( "6:argv --> %p\n", &argv );
    /* 7 */ (void) printf( "7:d --> %p\n", &d );
    /* 8 */ (void) printf( "8:e --> %p\n", &e );
    /* 9 */ (void) printf( "9:f --> %p\n", &f );
    /*10 */ (void) printf( "10:malloc --> %p\n", malloc(d) );
    return 0;
}
```

This line number would print the smallest value/address

This line number would print the largest value/address

What question would you like to see on the Final Exam? (1 pt)