1) Which part of the entire compilation sequence clear through to program execution is responsible for:
ensuring the bss segment is set up and zero-filled ________________________________
translating C source code into assembly target code ________________________________
translating assembly source code into object target code ________________________________
getting the executable image from disk into memory ________________________________
resolving undefined external references with defined global references across modules _________________
creating an executable image from multiple object files ________________________________

2) Specify the scope/visibility of each of the following:
external static variables _____
internal static variables _____
global functions _____
global variables _____
local variables _____
formal parameters _____
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( 2, 10 )? Hint: Draw stack frames.

```c
int recurse( int a, int b ) {
    int local = b - a;
    int result;
    if ( b > 7 )
        result = local + recurse( a, b - 1 );
    else
        result = local;
    printf( "%d\n", result );
    return result;
}
```

(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?

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

int a;

void foo( int b )
{
    static int c = 42;
    int d = 420;
    /* 1 */ (void) printf( "1:d --> %p\n", &d );
    /* 2 */ (void) printf( "2:a --> %p\n", &a );
    /* 3 */ (void) printf( "3:b --> %p\n", &b );
    /* 4 */ (void) printf( "4:c --> %p\n", &c );
}

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