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

a) getting the executable image from disk into memory ________________________________

b) ensuring the bss segment is set up and zero-filled ________________________________

c) creating an executable image from multiple object files ________________________________

d) translating C source code into assembly target code ________________________________

e) resolving undefined external references with defined global references across modules ________________________________

f) translating assembly source code into object target code ________________________________

Order the following storage elements/types from fastest to slowest

A) L1 cache
B) RAM (Main Memory)
C) Hard disk
D) Registers
E) Tape
F) L2 cache

_____ (Fastest)

_____  

_____  

_____  

_____  (Slowest)

#2. What gets printed if the following function is invoked as `recurse( 4, 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 );
}
```

Put answers here
#3. 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;

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

int main( int argc, char *argv[] ) {
    static int c = 420;
    int d;
    int e = 42;
    /* 2 */ (void) printf( "2:c --> %p\n", &c );
    /* 3 */ (void) printf( "3:argc --> %p\n", &argc );
    /* 4 */ (void) printf( "4:foo() --> %p\n", foo );
    /* 5 */ (void) printf( "5:e --> %p\n", &e );
    foo();
    /* 6 */ (void) printf( "6:malloc --> %p\n", malloc(e) );
    /* 7 */ (void) printf( "7:d --> %p\n", &d );
    /* 8 */ (void) printf( "8:argv --> %p\n", &argv );
    /* 9 */ (void) printf( "9:a --> %p\n", &a );
    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)