#1. Given

```c
static void fubar( int a )
{
    void (*b) (int) = fubar;
    static int c = 5;
    ...
}
```

When this function is called, identify which area of the C Runtime Environment each of the following will be allocated and its scope or visibility.

<table>
<thead>
<tr>
<th>Area of Runtime Env.</th>
<th>Scope/Visibility (Global/File/Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>fubar</td>
<td></td>
</tr>
</tbody>
</table>

Where b is pointing ____________

If the function above is called 15 times, indicate how many times will c be initialized to 5? _________

Using the Rt-Lt Rule, define a variable named bar that is a pointer to an array of 13 elements where each element is a pointer to a struct foo.

#2. a) Convert $111.75_{10}$ to binary fixed-point and single precision IEEE floating-point representation (expressed in hexadecimal).

- binary fixed-point __________________________________ $\times 2^0$
- IEEE floating-point __________________________________ (hexadecimal)

b) Convert $0xC20E8000$ (single precision IEEE floating-point representation) to fixed-point decimal.

- fixed-point decimal __________________________________ (decimal / no exponential notation)
#3. What is the output of the following program? (Hint: Draw stack frames!)

```c
int
main()
{
    int a = 4;
    int b = 9;
    swap1( &a, b );
    printf( "%d\n", a );
    printf( "%d\n", b );
    a = 7;
    b = 13;
    swap2( a, &b );
    printf( "%d\n", a );
    printf( "%d\n", b );
    return 0;
}
```

#4. What gets printed with the function call `mystery( 1 );`? (Hint: Draw stack frames!)

```c
int mystery( int param )
{
    int local = 4;
    if ( local > param )
    {
        local = local + param;
        printf( "%d\n", local ); /* Output the value of local followed by a newline */
        param = mystery( local - 2 ) + local;
        printf( "%d\n", param ); /* Output the value of param followed by a newline */
    }
    else {
        printf( "Stop\n" );
    }
    return local;
}
```