#1. In the SPARC architecture, to access local variables allocated on the stack you would use a ____________
(positive or negative) offset relative to register ____________.

The _________________ SPARC instruction saves the current value of %pc into %o7.

The _________________ SPARC instruction adds ____ to the value in %i7 and sets %pc with the result.

Using the Rt-Lt Rule, define a variable named foo that is a pointer to a function that takes a pointer to a double
as its single argument and returns a pointer to a struct fubar.

---

#2. a) Convert $97.875_{10}$ to binary fixed-point and single precision IEEE floating-point representation
(expressed in hexadecimal).

binary fixed-point ______________________ x $2^0$

IEEE floating-point ___________________________ (hexadecimal)

b) Convert $0xC27E8000$ (single precision IEEE floating-point representation) to fixed-point decimal.

fixed-point decimal ___________________________ (decimal / no exponential notation)
#3. Given

```c
int a = 420;
static void fubar( int b )
{
    static int c;
    int *d = &b;
    ...
}
```

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>d</td>
<td></td>
</tr>
<tr>
<td>fubar</td>
<td></td>
</tr>
<tr>
<td>Where d is pointing</td>
<td></td>
</tr>
</tbody>
</table>

If the function above is called 5 times, indicate how many times will d be initialized to &b? ______

#4. What gets printed with the function call `mystery(11)`? (Hint: Draw stack frames!)

```c
int mystery( int param ) {
    int local = 5;

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

    return local;
}
```