#1. In the SPARC architecture, if the caller passes 4 actual arguments to a function, the values of those actual arguments are accessed by the callee in register ______ through _______. The _____________ instruction is used to allocate space for local variables on the stack. The callee places the return value in register ______ before executing the ______ / ______ instruction sequence.

Why is it a bad thing to return a pointer to a local variable or parameter?

Using the Rt-Lt Rule, define a variable named foo that is an array of 7 elements where each element is a pointer to a function that takes a pointer to a short as its single argument and returns a pointer to a struct bar.

#2. a) Convert 103.125₁₀ to binary fixed-point and single precision IEEE floating-point representation (expressed in hexadecimal).

binary fixed-point ____________________________ x 2⁰

IEEE floating-point ___________________________ (hexadecimal)

b) Convert 0xC2068000 (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 = 5;
    int b = 10;
    swap1( a, b );
    printf( "%d\n", a );
    printf( "%d\n", b );
    a = 15;
    b = 20;
    swap2( &a, &b );
    printf( "%d\n", a );
    printf( "%d\n", b );
    return 0;
}

void swap1( int a, int b )
{
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
}

void swap2( int *a, int *b )
{
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
```

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

```c
int mystery( int param )
{
    int local = 6;

    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( "Halt\n" );
    }

    return local;
}
```