

This next problem will test your understanding of stack frames. It is based on the following recursive C function:

```
int silly(int n, int *p)
{
    int val, val2;

    if (n > 0)
        val2 = silly(n << 1, &val);
    else
        val = val2 = 0;

    *p = val + val2 + n;

    return val + val2;
}
```

This yields the following machine code:

```
silly:
    pushl %ebp
    movl %esp,%ebp
    subl $20,%esp
    pushl %ebx
    movl 8(%ebp),%ebx
    testl %ebx,%ebx
    jle .L3
    addl $-8,%esp
    leal -4(%ebp),%eax
    pushl %eax
    leal (%ebx,%ebx),%eax
    pushl %eax
    call silly
    jmp .L4
    .p2align 4,,7
.L3:
    xorl %eax,%eax
    movl %eax,-4(%ebp)
.L4:
    movl -4(%ebp),%edx
    addl %eax,%edx
    movl 12(%ebp),%eax
    addl %edx,%ebx
    movl %ebx,(%eax)
    movl -24(%ebp),%ebx
    movl %edx,%eax
    movl %ebp,%esp
    popl %ebp
    ret
```

Problem 34. (6 points):

- A. Is the variable `val` stored on the stack? If so, at what byte offset (relative to `%ebp`) is it stored, and why is it necessary to store it on the stack?

- B. Is the variable `val2` stored on the stack? If so, at what byte offset (relative to `%ebp`) is it stored, and why is it necessary to store it on the stack?

- C. What (if anything) is stored at `-24(%ebp)`? If something is stored there, why is it necessary to store it?

- D. What (if anything) is stored at `-8(%ebp)`? If something is stored there, why is it necessary to store it?