Names: 

In this lab, we expand on our mental model of computation to consider how function parameters change our stack-model of computation.

**Review: Loops and Mental Models**

For each of the following programs, give the state of the stack and what has been outputted to the console at each of the commented program points. Make sure to reproduce your diagrams exactly for each program state.

```c
int main(void) {
    int result = 1;
    // Point (A)
    for (int i = 4; i >= 0; i = i - 2) {
        // Point (B)---first iteration only
        result = result + i * 2 % 3;
        printf("%d ", result);
        // Point (C)---final iteration only
    }
    // Point (D)
    return 0;
}
```
Function Parameters

With respect to our mental model of computation, we can think of function parameters as local variables stored in the stack frame of that function. These variables are initialized with copies of the arguments passed to the function. Otherwise, they behave identically to locals—in particular, when a function returns, the function’s stack frame is popped by the stack.

For each of the following programs, give the state of the stack and what has been outputted to the console at each of the commented program points. Make sure to reproduce your diagrams exactly for each program state.

```c
int compute(int x, int y, double d) {
    // Point (A), (C)---both calls
    return (int) (x * d + y * d);
}

int main(void) {
    int ret = 0;
    ret = compute(3, 5, 2.0);
    // Point (B)
    compute(1, 10, -1.0);
    // Point (D)
    return 0;
}
```
void change(int x) {
    // Point (B)
    x = 10;
}

int main(void) {
    int x = 0;
    // Point (A)
    change(x);
    // Point (C)
}

int compute(int n) {
    // Points (A), (B), and (C)---every call to compute
    if (n > 0) {
        return compute(n - 1) * n;
    } else {
        return 1;
    }
}

int main(void) {
    int ret = compute(3);
    // Point (D)
    return 0;
}
The Evil of Globals

Finally, global variables are variables declared outside any particular function. In terms of our stack model of computation, they exist in a separate part of memory called the static region which you can draw in your diagrams as a collection of variable-value pairs separate from the stack. They are evil and you should not use them, instead favoring explicitly passing data between function calls using parameters and return values. To see why, work through this tracing problem.

```c
int glob = 10;

void g() {
    // Point (B)---first call to g()
    for (int i = 0; i < glob; i++) {
        printf("%d ", i);
    }
    printf("\n");
    glob += 1;
    // Point (C)---final call to g()
}

int f() {
    for (int i = 0; i < glob; i += 2) {
        // Point (A)---first iteration of the for-loop
        g();
        printf("===\n");
    }
    return glob;
}

int main(void) {
    int ret = f();
    // Point (D)
}
```