**Exercise 2**: Begin at the activation record for the function that is nested *n* levels deep. Follow the nesting link in the activation record to move to the activation record of the surrounding block. You want to move up *n*−*m* levels. Now you will be at the activation record for the function nested *m* levels deep. Now, within this activation record, you can locate the desired variable.

If *m*>*n*, it means the function making the reference is less deeply nested than the function where the variable is defined, and this is not a legal reference.

**Exercise 3**:

* **m < n**: The new activation record's nesting link should point to the current activation record because the new function is directly enclosed by the current function.
* **m = n**: The new activation record's nesting link should point to the same place as the current activation record's nesting link.
* **m > n**: The new activation record's nesting link should point to the activation record of the function that directly encloses the new function being called. To find this, you need to traverse the nesting links of the current activation record m-n times.

A function can never directly call another function that is nested more than one level deeper than itself, because it wouldn't have direct knowledge of that function.

**Exercise 7:**

1. Yes, there are no ways for anything to reference its local variable after the function finishes running
2. No, the closure that is returned needs to be able to access the local variable x in the function even after it returns the closure.
3. Yes, the function doesn’t use any local variables that you would need the activation record to try and reference.
4. Yes, there are no ways for anything to reference its local variable after the function finishes running

**Exercise 8:**

1. No, there are no local variables in the surrounding block that function f would need the nesting link to access
2. Yes, function f would need to use the nesting link to access the local variable n
3. Maybe, it was not defined in the function do123 and we do not know where it was defined.

**Exercise A:** Local variables are only needed while the function is running making space for these variables on the stack allows for more efficient memory management

**Exercise B:** a failure would not occur here

**A screenshot of a graph

Description automatically generated**

**A failure would occur here and it could be fixed by coalescing the problem**

**A screenshot of a math game

Description automatically generated**

**A failure would occur and it could NOT be fixed by coalescing**

**A screenshot of a graph

Description automatically generated**