1. a) **Answer:** 9 4 2 1 4.

b) **Answer:**

![Diagram of stack frames and links]

The diagram illustrates the stack frames and links for variables A, B, and M. Each node represents a block, and variables are assigned locations immediately following the variables of the surrounding (parent) block. Variables of siblings overlap.

c) **Answer:** It dereferences its static link to find the stack frame of B. Within this frame it finds B’s static link at a statically known offset. It dereferences that to find the stack frame of main, within which it finds g, again at a statically known offset. (This assumes that g is a local variable of main, not a global variable.)

2 **Answer:** In the code above, variables a, b, and c are live throughout the execution of the outer block. Variables d, e, and f are needed only in the first nested block, and can overlap the space devoted to g, h, and i. A total of $4 \times 6 = 24$ bytes is required.

When compiling a subroutine, the compiler can construct a tree in which each node represents a block, and is a child of the node that represents the surrounding block. Variables declared in the outermost block are assigned locations at the beginning of the subroutine’s space. Variables in a nested block are assigned locations immediately following the variables of the surrounding (parent) block. Variables of siblings overlap.

3 **Answer:** Since Fortran 77 lacks recursion, the call graph for a Fortran 77 program is guaranteed to be acyclic. If two subroutines never appear on the same path from main in this graph, then their local variables will never be needed at the same time, and may in fact share space. If our compiler allocates local variables statically, we can choose a minimum-size allocation by performing a topological sort of the call graph. The “frame” of any subroutine that can be called only from the
main program is placed at address 0. After this, the frame of any subroutine that can be called only by subroutines whose frames have already been assigned locations is placed at the first address that is beyond the frames of all of its potential callers.

As an example, consider the acyclic call graph below. It shows that subroutine E is never active at the same time as B or C. When allocating space we can therefore overlap E’s storage with that of B and C. Similarly, the space for D and F can overlap, though both must be disjoint from A and E. (The vertical offsetting of boxes is for clarity of presentation only.)

4. **Answer:** (a) 3; (b) 4; (c) 1.