Problem 1 Ken Thompson’s paper, “Reflections on Trusting Trust,” describes a technique for installing an undetectable login backdoor by adding a second backdoor to the compiler. The backdoored compiler inserts the appropriate backdoors when compiling the login program and the compiler itself. Once the binary of the compiler, used for bootstrapping future systems, implements the backdoor, any trace of tampering can be removed from the source. In this problem, we will explore a technique for detecting such an attack.

Assume we have two C-language compilers: GCC and Clang. We suspect that nefarious hackers have inserted the Thompson backdoor into the GCC binary on our Linux system, but we believe that these hackers are haven’t managed to corrupt the faculty and students at UIUC responsible for Clang. The entire Linux system doesn’t yet build with Clang, since many programs were written to expect GCC’s extensions to the C language. But we can get Clang to build GCC. (The usual procedure is to compile a new version of GCC using GCC itself; this is known as “self-hosting.”)

Describe how we can reliably detect the presence of a GCC backdoor using the fact that we can compile GCC with Clang.

In this problem, you have access to a Linux runtime environment in which the GCC compiler is possibly compromised, but you may assume that other critical components (such as the kernel, shell, and standard utilities) are clean.
Problem 2 Suppose you are driving along the highway and see the sign pictured below:

Where will the police actually be waiting? Explain. It may help to explain the threat model you are assuming.

Hint: Why put a sign up at all?