The goal of this assignment is to integrate an interpreter and a true JIT compiler into a single executable that initially interprets each method, but JIT-compiles it after it has been executed 3 times, and thereafter runs the compiled code. You are provided with a working interpreter (a minor variation of the solution to homework 1) and a true JIT compiler (a modification of the solution to homework 3), which delays compilation until just before a method is first executed. The two share a common set of runtime routines runtime.ch; this serves to avoid redundancy and to guarantee consistent array layout conventions.

Your combined system should behave as follows:

- **When interpreting a call:**
  - If the target method has already been compiled, call the compiled code.
  - If not, but the method has already been called 3 times, compile it now, and then call the compiled code.
  - Otherwise, call the interpreter recursively to execute the method in the existing way.

- **When compiling a call:**
  - If the target method has already been compiled to native code or has a trampoline, generate a call to that code.
  - Otherwise, compile a trampoline for the target method, and generate a call to that trampoline.

- **Trampolines should check if the target method has already been called 3 times.**
  - If so, they should compile it now, rewrite themselves (and the caller) and call the compiled code in the existing way.
  - If not, they should interpret the target method.

Note that the main method should always be interpreted.

As in the existing JIT, you should ensure that once a method has been compiled to native code, all subsequent calls to that method invoke the native code as expeditiously as possible. Remember to consider, and generate java test cases for, all possible combinations of caller/callee state (interpreted or compiled).

As noted in class, the trickiest part of the existing JIT code is that it when the trampoline compiles a method it updates its own caller to call the resulting native code directly next time it is executed. Note that this is the wrong thing to do when calling from the interpreter! The easiest way to avoid problems is to maintain an invariant that the interpreter never calls into a trampoline—only into the body of a compiled method. That way, you don’t need to mess with the tricky trampoline code at all. But be sure you understand the consequences of maintaining this invariant!
How to submit your homework

Submit the homework *by email* to cs577apt@cs.pdx.edu prior to the beginning of class on the due date. You should submit a revised version of files jit.c and interp.c, together with any other new or changed files, as *attachments* to your mail.