1. **Java Interface Dispatch**

Consider the following two interface dispatch schemes, as described in the assigned paper by Alpern, et al.:

(i) Directly Indexed ITables (section 4.1, para. 2)

(ii) Interface Method Tables with 5 entries (IMT-5) (section 5)

(a) Characterize a set of Java programs that cause method (i) to use large amounts of space for data structures, but that method (ii) handles well. Explain why.

(b) Conversely, characterize a set of Java programs that cause method (ii) to use large amounts of space and time for conflict resolution stubs, but that method (i) handles well. Explain why.

(c) According to the authors, how did the performance of these two schemes compare on real programs?

(d) [for discussion] How might the idea of speculative specialization be applied in other areas of programming language implementation?

2. **.NET Generics**

Consider the assigned paper by Kennedy and Syme.

(a) Describe a significant difference, having to do with types, between JVM bytecode and the original (non-generic) .NET IL.

(b) What does the paper mean by “code specialization” and “code sharing” and what are the trade-offs between them?

(c) [for discussion] Assess the quality of the paper’s performance section. How might it be improved?