6.189 Multicore Programming Primer **Mini-Quiz #4 (1/19/2007)**

In 1968, ACM published Edsger Dijkstra's letter titled *Go To Statement Considered Harmful*. Dijkstra argued that GOTO statements should be abolished because they lead to unstructured control flow and complicate the analysis and verification of programs. In 1974, Don Knuth presented an alternative viewpoint in *Structured Programming with go to Statements.* He showed that for some common programming tasks, GOTOs are the best language construct to use. This was more than 30 years ago. Today (and looking into the future) the programming landscape is changing because of the new software crisis. How would *structured* vs. *unstructured* programming help programmers more rapidly express their parallel computation? What are some of the tradeoffs they might experience?

New applications have to exploit the multicore parallelism if they are to run faster on emerging processors. This means that applications have to explicitly describe their parallelism, or compilers have to extract parallelism with practical efficiency. Either of these requirements is more easily satisfied with a structured programming approach.

In streaming applications for example, structured dataflow graphs readily expose communication and computation patterns, thereby allowing compilers to more readily adjust the granularity of the computation for a load balanced execution. An unstructured dataflow graph may obscure communication patterns, making it harder to determine synchronization barrier, leading to unbalanced execution.

The structured programming approach however may force programmers to coerce dataflow patterns to fit the structure allowed by the programming model, and the resultant code may not be as naturally appealing had the language allowed for more unstructured communication patterns. In StreamIt for example, splitters and joiners are required to appear within single-input-single-output splitjoins, whereas they might be useful as first class stream elements (i.e., used outside of splitjoins).