Architecture
- ARC.1: I can explain and distinguish between the architectural concepts of pipelined, shared memory (threaded), and distributed memory (message passing) architectures.
- ARC.2: I can explain and distinguish between network topologies used in parallel architectures and how topologies can affect parallel program performance.

Efficiency
- EFF.1: I can use appropriate techniques to compute speedup and efficiency given timings for the execution of a parallel program.
- EFF.2: I can use appropriate techniques to compute the optimal speedup possible for a program and I can compare measured program performance against the optimal.
- EFF.3: I can compare and contrast the efficiency of alternative parallel algorithms and implementations of a program given code.

Critical Sections
- CRI.1: I can determine critical sections within code and suggest how mutual exclusion techniques can be used to remove possible errors based on critical section conflicts.
- CRI.2: I can choose mutual exclusion techniques suitable for a given parallel programming problem and API and be able to justify my choice.

Threading and Message Passing
- TMP.1: I can choose between threading and message passing for a given parallel programming problem and be able to justify my choice.
- TMP.2: I can choose between point-to-point and collective communication functions within a message passing context and justify my choice.
- TMP.3: I can compute the amount of data transferred for parallel programs employing message passing.

Parallel Programming APIs
- API.1: For multiple parallel programming APIs, I can read code using parallel programming primitives, explain what that code is doing, and reason about the contents of memory after operations have been performed.
- API.2: I can profile code, interpret code profiling outputs, and suggest changes to parallel programs based on code profiling.
- API.3: I can suggest and provide a rationale for how to attack a problem using multiple parallel programming APIs.
- API.4: I can demonstrate understanding of the purpose and underlying implementations of high-level functionality provided by parallel programming APIs.
- API.5: I can correctly implement timing function code to measure the performance of a parallel program.
- API.6: I can make use of appropriate tools to compile and debug programs using parallel programming.
Parallel Program Design

• DES.1: I can suggest and provide a rationale for how to attack a problem via data parallelization, task parallelization, and pipelining approaches.
• DES.2: I can suggest and justify distribution of parallel work under threading, message passing, loop iteration scheduling, and pipelining models.
• DES.3: I can suggest when and where barriers should be used in designing a parallel programming solution to a problem and justify their use and placement in a program using parallel programming techniques.
• DES.4: I can recognize data dependencies within a program, be able to explain how those data dependencies may affect parallel execution of a program, and be able to suggest parallel programming designs that deal appropriately with data dependencies allowing for correct program execution.

Parallel Programming

• PRO.1: I can correctly implement solutions to simple-to-medium complexity problems using parallel programming with threads.
• PRO.2: I can correctly implement solutions to simple-to-medium complexity problems using parallel programming with message passing.
• PRO.3: I can correctly implement solutions to simple-to-medium complexity problems using parallel programming with OpenMP.