*Exercises:*

* [loop\_ex1.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex1.c) – parallelizing loops by combining the parallel and for worksharing constructs.
* [loop\_ex2.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex2.c) – an example of parallelizing a serial loop by first modifying the loop to ensure all iterations are independent.
* [loop\_ex3.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex3.c) – an example of two parallel loops in the same parallel block that also employs the nowait clause.
* [loop\_ex4.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex4.c) – parallelizing nested loops with the collapse clause.
* [loop\_ex1\_if\_clause.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex1_if_clause.c) – an if clause can be added to a parallel construct and allows for code to be parallelized under some conditions (e.g., a high number of threads) and not parallelized under other conditions (e.g., a low number of threads).
* [loop\_ex2\_ordered\_clause.c](http://www.sqrlab.ca/exercises/csci4060u-w17/loop_ex2_ordered_clause.c) – an ordered clause can be used with a parallel for loop to order the output of the loop the same as if it was executed sequentially. The ordered clause does have a performance cost.
* [single\_ex1.c](http://www.sqrlab.ca/exercises/csci4060u-w17/single_ex1.c) – ensures a block of code is executed by only one thread with an implicit barrier afterward.
* [master\_ex1.c](http://www.sqrlab.ca/exercises/csci4060u-w17/master_ex1.c) – ensures a block of code is executed by the master thread with no implicit barrier afterward.
* [sections\_ex1.c](http://www.sqrlab.ca/exercises/csci4060u-w17/sections_ex1.c) – allows for different blocks (sections) of code to be executed by different threads (each with an implicit barrier afterward).