# CS 320/CSE 302/ECE 392

Introduction to Parallel Programming for Scientists and Engineers
MP3 - Due Monday, March 16

Implement in OpenMP the blocked version of the cyclic reduction algorithm presented in class. This algorithm is a generalization of the blocked algorithm presented in class for parallel prefix. Notice that the blocked parallel prefix presented in class is a vector algorithm and for this exercise we are asking for an OpenMP version. In fact, although in MP2 we asked for a direct translation of each Fortran 90 operation into OpenMP, in general is not a good idea to first develop a vector algorithm and then do a direct translation.
Thus, if you wanted to develop a parallel prefix program in OpenMP the translation should not be a direct one. In particular, the loop

```
          do i=2, ⌈n/P⌉
              C(i,1:P) = C(i-1,1:P) + C(i,1:P)
          end do
```

should be represented in OpenMP as follows

```
c$omp parallel do
          do j=1,P
              do i=2, ⌈n/P⌉
                  C(i,j) = C(i-1,j) + C(i,j)
              end do
          end do
```

This loop accomplishes the same thing as the vector loop, but is more efficient than the doubly-nested loop with an inner parallel loop.

For the bonus 50 points, you should develop a Fortran 90 algorithm for cyclic reduction that follows not the array order of vectors *a* and *b*, but the order indicated by a vector of pointers.

As usual, solutions should be left in your home directories under the name **mp2.f** and **mp2.bonus.f**.