Instructions: Answer as many of problems 1, 2, 3, and 4 as you can. Answer at most one of problem 5 or 6. If you answer both problem 5 and problem 6, an arbitrary one will be graded.

1. (20 points) Show that the clique problem is self-reducible. The decision problem is to take a graph $G$ and an integer $k$ and decide if $G$ has a clique of size $k$ or not. The optimization problem takes a graph $G$, and returns a largest cardinality clique in $G$. Recall that a clique is a collection of mutually adjacent vertices.

Start by explaining what it means for a problem to be self-reducible. I am interested in knowing both whether you know how to show a problem is self-reducible in general, and whether you know how to show that the clique problem is self-reducible in particular.

2. (20 points) For each of the following two problems, either prove that it is NP-hard by reduction (using the clique decision problem, or the independent set decision problem), or give a polynomial time algorithm. The clique decision problem is to take a graph $G$ and an integer $k$ and decide if $G$ has a clique of size $k$ or not. Recall that a clique is a collection of mutually adjacent vertices. The independent set decision problem is to take a graph $G$ and an integer $k$ and decide if $G$ has an independent set of size $k$ or not. Recall that an independent set is a collection of mutually nonadjacent vertices. For NP-hardness arguments, make sure it is clear that state the direction of the reduction, the whole reduction (including those parts that are generic), and what properties one needs of the reduction.

(a) The input is an undirected graph $G$ and an integer $k$. The problem is to determine if $G$ contains a clique of size $k$ AND an independent set of size $k$.

(b) The input is an undirected graph $G$ and an integer $k$. The problem is to determine if $G$ contains a clique of size $k$ OR an independent set of size $k$.

3. (20 points) We consider the problem of multiplying two $n \times n$ matrices. Assume that the sequential time algorithm that you wish to compare to has time complexity $S(n) = n^3$.

(a) Design a parallel algorithm that runs in time $O(\log n)$ time on a CREW PRAM with $n^3$ processors.

(b) What is the efficiency of this algorithm? Start with a definition of efficiency.

(c) Using the folding principle, what upper bound would you get on the running time for this algorithm on $n^{1/3}$ processors? Start by stating the folding principle.

(d) Explain how to modify this algorithm to get a parallel algorithm that runs in time $O(\log n)$ time on a EREW PRAM with $n^3$ processors.

4. (20 points)

(a) Explain how to merge two sorted arrays of $n$ numbers in time $O(\log n)$ on an EREW PRAM with $n$ processors. Give a short justification that the time is $O(\log n)$.

(b) Explain how to use the above procedure to sort $n$ numbers in $O(\log^2 n)$ on an EREW PRAM with $n$ processors. Give a short justification that the time is $O(\log^2 n)$. In principle, you can answer this part of the question, without answering the previous part.
Remember, answer at most one of problem 5 or 6. If you answer both problem 5 and problem 6, an arbitrary one will be graded.

5. (20 points) In the disjoint paths problem the input is a directed graph $G$ and pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of vertices. The problem is to determine if there exist a collection of vertex disjoint paths between the pairs of vertices (from each $s_i$ to each $t_i$). Show that this problem is $NP$-hard by a reduction from the 3SAT problem.

6. (20 points) Design a parallel algorithm that finds the maximum number in a sequence $x_1, \ldots, x_n$ of (not necessarily distinct) integers in the range 1 to $n$. Your algorithm should run in constant time on a CRCW Common PRAM with $n$ processors.