Quadratic programming: Prove that the following problem is NP-hard:

Quadratic programming: Given a set of equations $E_1, \ldots, E_t$ in $n$ variables $x_1, \ldots, x_n$ of the form: $E_k$ is $\sum_{i=1}^{n} a_{i,k} x_i^2 + \sum_{1 \leq i \leq j \leq n} b_{i,j,k} x_i x_j + \sum_{1 \leq i \leq n} d_{i,k} x_i = c_k$ where $a_{i,k}, b_{i,j,k}, d_{i,k}, c_k$ are given integers. Decide if there is a solution, i.e., an assignment of real numbers $v_1, \ldots, v_n$ to the variables making all equations true. Is this problem in $NP$? (Don’t give a formal proof for whether the problem is in NP) (Hint: consider the equation: $x_i^2 = x_i$.)

Co-NP Assume $NP \neq Co-NP$. Prove from this that there is a language $A \in P^{NP}$ so that $A \notin NP \cup co-NP$.

NP-Completeness We say that $NP$ complete problems are the “hardest” in $NP$, but intuitively that means they are the “least likely to be easy”
not that they have the greatest worst-case complexity. To illustrate this, prove that for every $k > 0$ there is an $NP$–complete language $L$ so that $L \in TIME(2^{n^{1/k}})$.

**Polynomial-time hierarchy** Assume that $NP \subseteq TIME(n^{O((\log n)^i)})$. Prove that $\Sigma_i \subseteq TIME(n^{O((\log n)^{2^{i-1}})})$.

**Sudoku experiment** We started looking at reducing sudoku problems to SAT last assignment.

The *sudoku* problem of size $n$ is as follows. The input is an $n^2 \times n^2$ matrix $M$ whose entries are either “blank” or an integer between 1 and $n^2$. A solution fills in the blank spaces with integers between 1 and $n^2$. The following constraints must be met: Each integer from 1 to $n^2$ appears exactly once in each row, in each column, and in each $n \times n$ sub-matrix of the form $M[jn + 1\ldots(j + 1)n][in + 1\ldots(i + 1)n]$ for each $0 \leq i, j \leq n - 1$. The problem is to find any solution meeting the constraints, or return “no solution possible” if there is no such solution.

Last assignment, you gave at least two different ways to reduce the Sudoku problem to $CNF – SAT$.

Try solving sudoku problems by combining the above reductions with a complete SAT solver, such as Zchaff (download page, //www.princeton.edu/~chaff/zchaff/index2.html)

Use the following test distribution: For each diagonal block, $M[in + 1\ldots(i+1)n][in+1\ldots(i+1)n]$, fill in the block with a random permutation of the integers from 1..$n^2$. Leave other blocks blank.

The experiment should return the following information: For as many values of $n$ as you can in a reasonable amount of time, what fraction of such puzzles have solutions? (You may not always be able to tell, if the SAT solver does not terminate in a reasonable amount of time.) How much time did the SAT solver take using different reductions? Which reduction is best?

(Note: Be careful not to use up too much computer time. Don’t leave programs running unsupervised too long. Depending on your algorithm and reduction, you may find even very small sizes take huge amounts of time. However, some credit will be based on getting results for larger $n$.)