1. [Make a truth table. The first, second, third, fifth, and seventh rows are true, in my ordering. Each row corresponds to a clause in the following proposition.] The given proposition is logically equivalent to

\[(p \land q \land r) \lor (p \land q \land \bar{r}) \lor (p \land \bar{q} \land r) \lor (\bar{p} \land q \land r) \lor (\bar{p} \land \bar{q} \land r).\]

[This answer is not unique.]

2A. TRUE. [The product of \(a/b\) and \(c/d\) is \((ac)/(bd)\).]

2B. FALSE. [Addition of \(n \times n\) matrices must be \(\Omega(n^2)\), because \(n^2\) entries must be computed.]

2C. FALSE. [There is a simple algorithm, that iteratively computes \(F_0, F_1, \ldots, F_n\), in time \(\Theta(n)\). Unfortunately, the question was worded somewhat ambiguously. I gave one point to an incorrect answer.]

2D. FALSE. [The proposition says that there is a person \(y\) such that for all people \(x\), \(x\) is the mother of \(y\). In other words, there is a person \(y\) who is the child of everyone. That is false.]

2E. FALSE. [The product of the two matrices is \[
\begin{bmatrix}
20 & 14 \\
56 & 41
\end{bmatrix}.
\]

3. [By the way, this problem is taken from Assignment F, almost verbatim.] We will prove, using mathematical induction, that for all \(n \geq 0\), \(\sum_{k=0}^{n} kk! = (n+1)! - 1\).

Base case: Assume that \(n = 0\). Then \(\sum_{k=0}^{n} kk! = 0! = 0 = 1! - 1 = (n+1)! - 1\), as desired.

Inductive case: Assume that \(\sum_{k=0}^{n} kk! = (n+1)! - 1\). We wish to show that \(\sum_{k=0}^{n+1} kk! = (n+2)! - 1\). Well,

\[
\sum_{k=0}^{n+1} kk! = (n+1)(n+2)! + \sum_{k=0}^{n} kk!
\]

\[
= (n+1)(n+2)! + (n+1)! - 1 \quad \text{(by the inductive hypothesis)}
\]

\[
= (n+2)(n+1)! - 1
\]

\[
= (n+2)! - 1.
\]

This completes the inductive case and the proof by mathematical induction.

4A. Let \(T(n)\) be the time taken to sort a list of length \(n\). Then \(T(1) = c\) for some \(c\), and \(T(n) = 2T(n/2) + cn\), because sorting a list of length \(n\) involves:

- computing a median (time \(cn\)),
- scanning the list to split it into two sublists (time \(cn\)),
- making two recursive calls on lists of length \(n/2\),
- rejoining the lists (constant time or time \(cn\), depending on the data structure).

4B. Applying the master theorem of divide-and-conquer algorithms with \(a = 2\), \(b = 2\), \(k = 1\), we have \(b^k = a\), and hence \(T(n) = \Theta(n \log n) = \Theta(n \log n)\). [By the way, mystery sort is actually quick sort.]

5. Here is the `powerSet` function in Python:
def powerSet(l):
    if len(l) == 0:
        return [[]]
    else:
        notUsingHead = powerSet(l[1:])
        usingHead = [[l[0]] + part for part in notUsingHead]
        return notUsingHead + usingHead

[Some students’ answers used the comb function from homework, like this:

def powerSet(l):
    result = []
    for k in range(len(l) + 1):
        result.extend(comb(l, k))
    return result

Such an answer would ideally include code or explanation of how comb works. Otherwise, the existence of comb somewhat trivializes this problem, which was intended to be doable because of the comb problem but not trivial.]