Discussion Problems: "SAT Basics"
Declarative Methods (JHU 601.342/642)
Prof. Eisner

1. In a small town, Ms. Baker, Mr. Gardner and Ms. Mayer are the baker, the gardener, and the mayor—but not necessarily in that order. In fact, Mr. Gardner often remarks to the mayor: "How funny that all of us avoided our own obvious professions, yet ended up in one another's!"

   Rather than determining for yourself who has what job, write down the implicit constraints as a short SAT program. Explain what each variable means. There should be a unique satisfying assignment.

2. (a) [3 points] Convert this SAT problem to CNF, without using any additional variables:

   $$(A \wedge B) \vee (C \wedge (D \vee E \vee F))$$

   (b) [3 points] Convert the same SAT problem to CNF, this time using switching variables. This should give a shorter solution.

   (c) [8 points] How many satisfying assignments are there for each of the following formulas? (For example, the second formula has 3 variables: of the 2^3 possible assignments to those variables, how many are satisfying?)

   Note that these are "simple" formulas in which every variable is only used once, so it should be possible to build up your answer somehow from the sub-formulas.

   - _____ $(A \wedge B)$
   - _____ $(D \vee E \vee F)$
   - _____ $(C \wedge (D \vee E \vee F))$
   - _____ $(A \wedge B) \vee (C \wedge (D \vee E \vee F))$ [tricky!]

   (d) [3 points] It turns out that the formula that answers question 2b has more satisfying assignments than the original $(A \wedge B) \vee (C \wedge (D \vee E \vee F))$ did. Explain how this is possible, seeing as they express the same SAT problem!

   (You might perhaps find it helpful to think about how many satisfying assignments the formula in question 2b has, though take care to note that it is not a "simple" formula.)

3. *Background:* When studying SAT, we considered a constraint that could be written as `atleast(13, [A,B,...Z])`, which required that at least 13 of the boolean variables `A`, `B`, ... `Z` should be true. We saw in class how to encode this kind of constraint in CNF in polynomial space, by introducing some additional helper variables.

   Your problem today: Today, you will similarly consider the constraint `even([A,B,...])`, which requires that exactly an even number of the boolean variables `A`, `B`, ... should be true.

   (a) It is again possible to encode this in polynomial space in CNF. By "polynomial space," I mean that if there are n variables listed in the constraint `even([A,B,...])`, then the total length of the CNF clauses that encode this constraint is $O(n^k)$, for some constant k.

   Figure out what such a scheme would look like, and illustrate it for the $n = 3$ case by encoding `even([A,B,C])`. It should be completely clear from your answer what your general scheme is for all n. (That is, please write your answer out very clearly and systematically, as if it were generated by a *program* that could just as easily have encoded the $n = 26$ case, `even([A,B,...Z])`.)

   Note: If you prefer, the formula you give to encode `even([A,B,C])` does not actually have to be in CNF. It's okay to give *any* formula that uses only $\wedge, \vee, \neg, \rightarrow$. (Why? Because we know from class that any such formula *could* then be converted into CNF with at most polynomial blowup in the size of the formula, so that it is still $O(n^{k'})$ for some k'. You don't have to convert your formula to CNF yourself, as long as you know that this could be done! But your formula should not use $\leftrightarrow$ or xor, since those cannot be cheaply converted to CNF.)

   Hint to get you started: Our encoding in class of `atleast(13, [A,B,...Z])` was essentially recursive. Something recursive might work here, too. Notice for example that if `even([A,B,C,D,E,F,G])` is required and `G` is false, then `even([A,B,C,D,E,F])` must be required as well. This also suggests that "$\rightarrow$" might come in handy.

   (b) To encode an `even` constraint with n variables, how many additional variables do you need under your scheme, and how long is the total formula? Both answers are a function of n; give them in $O(\ldots)$ notation.