These practice problems are an opportunity for discussion and trying many different solutions. It is **not counted towards your grade**, and **you do not have to submit your solutions**. The purpose of these problems is to get more comfortable with reasoning and writing about NP completeness, approximation algorithms, and general grammars.

If you are stumped or looking for guidance, **ask**.

1. (Lewis & Papadimitriou 7.3.4) One way to come up with *new* NP-complete problems is to **generalize from a problem we already know is NP-complete**. Then, if certain parameters of the problem are fixed in a certain way, the problem in hand becomes a known NP-complete problem. One can reduce any problem to its generalization by simply introducing a new parameter, and otherwise leaving the instance as it is.

   Prove that the following language is NP-complete by showing that it is the generalization of an NP-complete problem. Give the appropriate parameter restriction.

   **LONGEST-CYCLE**: Given a graph $G$ and integer $k$, is there a cycle, with no repeated nodes, of length at least $k$?

   (Hint: what happens to this problem if $k$ is restricted to be equal to the number of nodes of the graph?)

2. Give a general grammar which generates the language:

   $\{a^{n^2} \mid n \geq 0\}$

   Prove that your grammar is correct.

3. Approximating TravelingSalesmanProblem.

   Recall the NP-complete decision problem TravelingSalesmanProblem, where we wanted to accept inputs that encoded a list of cities, the distance $d(u, v)$ between every pair of cities $u$ and $v$, and $k$ a limit on the total distance travelled, as long as it was possible for a salesman to visit every city, without repeats, and then return to her home city, in total distance $\leq k$.

   We now consider the optimization version of this problem, which is slightly modified:

   - we encode the cities as vertices of a graph $G$,
   - we encode the distances as labels on the edges of $G$,
   - the input does not include $k$, and
   - we want to output the smallest possible $k$ which is the distance of a roundtrip visiting every city exactly once, then returning home.

   For many TSP applications (such as when the cost is proportional to the distance between two cities), it makes sense for the edges to obey the triangle inequality: for every $i, j, k$, we have

   $$d_{(ik)} \leq d_{(ij)} + d_{(jk)}.$$  

   This version of the optimization problem is often called Metric-TSP.

   For this problem, we will develop a 2-approximation algorithm for Metric-TSP.
(a) First, to gain some intuition, consider the following graph:

(b) *On your own* try to identify a cheap tour of the graph.

(c) Build some more intuition by computing the minimum spanning tree (link: Wikipedia) of the graph. Let $T$ be your minimum spanning tree (MST).

(d) Let OPT be the cheapest tour. Show that its cost is bounded below by the cost of the MST: $\text{cost}(T) \leq \text{cost}(\text{OPT})$.

(e) Give an algorithm which returns a tour $A$ which costs at most twice the cost of the MST: $\text{cost}(A) \leq 2 \text{cost}(T)$.

(f) Conclude that your algorithm is a 2-approximation for Metric-TSP.