General rules for solving exercises

- This is a theory course, which means: if an exercise does not explicitly say “you do not need to prove your answer” or “justify intuitively”, then a formal proof is always required.
- All exercises and their solutions, no matter whether they are graded or regular/optional ones, are part of the material relevant for the two exams.
- The tasks marked with the symbol ⋆ are more difficult than usual exercises, either because they require a non-standard idea or involve some non-trivial calculation. This is merely a help for you to decide which exercises you would like to spend how much of your time on. Note that all exercises and their solutions count as part of the material relevant for the two exams.

Note: in the following exercises, we will refer to Theorem 7.6 that is going to be considered in next week’s lecture. You only have to read the statement in order to solve the following exercises.

Exercise 1 - PCP proof-checking version with ”larger” oracle verifiers

In this exercise, we will consider Theorem 7.6. We will show that it is not necessary to bound the length of the queries to w, because we can use the bound on the number of random coins.

Thus, assume that Theorem 7.6 holds, but w takes as input bit strings of length |C|^k for some constant k. Show that this implies Theorem 7.6, i.e., change the verifier such that you can replace w by w′ which takes as input strings of length O(log(|C|)).

Hint: How many different queries can be asked by the verifier?

Exercise 2 - PCP proof-checking version with adaptive proof querying

The proof-checking version of the PCP theorem in both the variants of Theorem 7.5 (“proof as a string”) and Theorem 7.6 (“proof as an oracle function”) are non-adaptive in accessing the proof in the following sense: The verifier computes all positions of the proof that it wants to access in advance, and only after that will access the proof at those very positions.

Let us now think about a more flexible type of verifier, which can compute the proof positions adaptively, i.e., it iteratively chooses a position q to access, and then depending on the outcome w(q) (being the content of the proof at q), it will compute the next position to access.

Show that for any such adaptive verifier accessing q_0 bits of the proof, there exists an equivalent non-adaptive verifier accessing q'_0 many bits for some constant q'_0, with both verifiers using the same number of random bits.

Exercise 3 - QP-SAT is NP-complete

A quadratic polynomial p in n variables over GF(2) can be written as p(x_1, ..., x_n) = a_0 + \sum_{i,j=1}^{n} a_{ij} x_i x_j, with a_0 and a_{ij} ∈ GF(2). The problem QP-SAT (satisfiability of quadratic polynomials) is defined as follows:

Given a collection p_1(x_1, ..., x_n), p_2(x_1, ..., x_n), ..., p_m(x_1, ..., x_n) of n-variate quadratic polynomials over GF(2), do there exist x_1, x_2, ..., x_n ∈ GF(2) with p_i(x_1, ..., x_n) = 0 for all i ∈ {1, ..., m}? (That is, do these polynomials have a common root?)

Show that the problem QP-SAT is NP-complete. In a possible proof you reduce 3-SAT to QP-SAT. Thus, you have to show that for any 3-CNF formula ψ you can in polynomial time generate a collection
of quadratic polynomials that have a common root if and only if $\psi$ is satisfiable. Exercises (a)-(c) provide a short guideline.

(a) For the clause $C = x_1 \lor \bar{x}_2 \lor x_3$ find a polynomial $p_C(x_1, x_2, x_3)$ (not necessarily of degree two) such that for $x_1, x_2$ and $x_3$, $p_C(x_1, x_2, x_3)$ evaluates to zero if and only if $C$ is satisfied.

(b) Replace $p_C(x_1, x_2, x_3)$ by two polynomials $q_C(x_1, x_2, x_3, y)$ and $r_C(x_1, x_2, x_3, y)$ of degree at most two and with one additional variable $y$ such that the roots of $p_C$ are the common roots of $q_C$ and $r_C$. (where the evaluation of $p_C$ is restricted to $x_1, x_2, x_3$).

(c) Generalize this idea for any clause of length three and finally construct, for a given 3-CNF formula $\psi$, a collection of quadratic polynomials such that they have a common root if and only if $\psi$ is satisfiable.

Exercise 4⭐ - Restricted Verifier Capabilities

(a) Prove the following statement: There exists a constant $\rho > 0$ such that for any language $L \in \text{NP}$, there exists a polynomial time verifier $V(x, w)$ with the following properties. The verifier expects a proof $w$ of size polynomial in $|x|$ for the statement $x \in L$. It first reads $x$, tosses $O(\log |x|)$ random coins, reads 3 bits of $w$, then accepts or rejects. If $x \in L$, then there exists a proof $w$ such that the verifier accepts with probability 1. If $x \notin L$, then for all $w$, the verifier rejects with probability at least $\rho$.

**Hint:** You may assume any of the theorems stated in the lecture notes (even the ones that were too hard for us to prove during the lecture).

(b) Somebody claims that there exists a constant $\rho > 0$ such that for any for any $L \in \text{NP}$, there exists a polynomial time verifier $V(x, w)$ with the following properties. The verifier expects a proof $w$ of size polynomial in $|x|$ for the statement $x \in L$. It first reads $x$, tosses $O(\log |x|)$ random coins, reads 3 consecutive bits (i.e., a substring $w_i w_{i+1} w_{i+2}$ for some $i$) of $w$, then accepts or rejects. If $x \in L$, then there exists a proof $w$ such that the verifier accepts with probability 1. If $x \notin L$, then for all $w$, the verifier rejects with probability at least $\rho$. Prove that if that somebody is right, then $P = \text{NP}$.

**Hint:** Reduce the problem to a satisfiability instance and make use of its specific properties.

(c) Prove: There exists a constant $\rho > 0$ such that for any language $L \in \text{NP}$, there exists a polynomial time verifier $V(x, w)$ with the following properties. The verifier expects a proof $w$ of size polynomial in $|x|$ for the statement $x \in L$. It first reads $x$, tosses $O(\log |x|)$ random coins, reads 3 contiguous bits and a $4^{th}$ separate one (i.e. a substring $w_i w_{i+1} w_{i+2}$ and some bit $w_j$ for some $i$ and $j$) of $w$, then accepts or rejects. If $x \in L$, then the verifier accepts with probability 1. If $x \notin L$, then the verifier rejects with probability at least $\rho$.

**Hint:** Adjust the proof format. (And: again, as in (a), you may assume all theorems stated in the notes)