EXERCISE ONE **Warmup with tokens.** Consider 10 tokens with values $1, 2, \ldots, 10$. If we split them into two groups – even valued and odd valued – the average of the odd group is 5 and the average of the even group is 6. Prove or disprove the following:

1. Can we reorder the tokens into two different groups so that the average of both groups increases?
2. Can we reorder the tokens into two different groups so that the average of both is above 5.5?

EXERCISE TWO **Card tricks.** An exercise heard at the lecture: we have 52 cards, half red, half black, randomly shuffled (a uniformly random permutation). We now reveal one card after another. Before each card you have the option of saying "I want the next card". Then we reveal the next card and you win if it is red, and the game ends if it is black. We are interested in the best algorithm, that is one maximizing the probability that it wins.

1. What is the probability of winning for the algorithm $Fi \equiv$ "always pick the first card"? And what is the probability for $La \equiv$ "always pick the last card"?
2. Let us search for a "better" algorithm compared to La in the following sense: We want to find an algorithm B for which it is true that with probability strictly greater than $1/2$, assuming that La pointed at a black card, B points at a red card. What is the major problem with such a notion of "better"?
3. Now let us return to the original notion of quality and consider the following algorithm that aims to be better than La: *"We keep revealing cards until we have more black cards revealed than red cards revealed. In such a situation, we stop: clearly in the deck there remain more red cards and our probability of picking the next card is strictly more than* $1/2$*. This situation is very likely to happen; after all, the expected number of red cards and black cards in the revealed section is* 0 *on even turns, so the color balance has to fluctuate around that. In the worst case when this never happens, we pick the last card, so we are at least as good as La."*
   Try to brainstorm a few informal/intuitive arguments that show or disprove that this new algorithm is better than La.
4. Our main and final task: find an algorithm which chooses the red card with probability strictly more than $1/2$ – or prove that no such algorithm exists.

EXERCISE THREE **Maximizing satisfiability.** In the maximization problem MAX-SAT we get on input a Boolean formula (expression) in the CNF form $((x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge \ldots)$. Throughout this exercise we will assume that every clause has at least two literals. Our task is not to check whether the entire clause is satisfiable, but to maximize the number of clauses that are satisfied.

1. Consider the following algorithm A: *"we take two assignments: one sets every variable to* 1*, one every variable to* 0*. We then return the assignment which satisfied more clauses."* What is the approximation ratio of this deterministic algorithm?
2. We strengthen A as follows: consider any algorithm B such that it tries constantly many assignments that are fixed before the input (similarly to A). It tries evaluating them all and it returns the best one. An assignment is any function $p : \mathbb{N} \to \{0, 1\}$, so it can be applied to any possible formula on input. What is the best approximation ratio that B achieves?
3. *Recall from the lecture:* What if we try a uniformly random assigment? Compute the expected number of satisfied clauses in this case.
4. Is MAX-k-SAT as hard as k-SAT? No, it is not. First let us verify that 2-SAT (on input we have a CNF formula with at most two literals per clause, and we have to check whether the formula is satisfiable) is polynomial-time solvable.
5. Show that MAX-2-SAT (On input we get a number c and a CNF formula with at most two literals per clause and we have to decide whether there is an assignment satisfying at least c clauses) is NP-hard.