- Start as early as possible, and contact the instructor if you get stuck.
- See the course outline for details about the course's grading policy and rules on collaboration.
- Submit your completed solutions to **Crowdmark**.

1. **Collapse of the polynomial hierarchy** [**6 marks**]
   (a) Show that if $\Sigma_k \supseteq \Pi_k$ or $\Sigma_k \subseteq \Pi_k$ then the polynomial hierarchy collapses.

   (b) Show that if $\Sigma_i = \Sigma_j$ or $\Pi_i = \Pi_j$ for $i < j$ then the polynomial hierarchy collapses.

2. **Complete problems for big classes** [**12 marks**]
   We introduced a definition of $\mathcal{C}$-complete languages for any class of languages $\mathcal{C}$ that contains P.
   (a) Recall $\mathsf{E} = \mathsf{TIME}(2^{O(n)})$, or "linear exponential time". Prove that there exists an E-complete problem.

   (b) Recall the complexity class $\mathsf{ELEMENTARY}$.

$$\mathsf{ELEMENTARY} = \bigcup_{k \geq 1} \mathsf{TIME}\left(\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}}}_{k}\right)$$

   Prove that there does not exist an $\mathsf{ELEMENTARY}$-complete problem.

   (c) Recall that RE is the set of languages recognizable by a Turing machine. Prove that there exists an RE-complete problem.

3. **Low-depth circuits for regular languages** [**8 marks**]
   The set of regular languages, REG, is contained in LIN because we can recognize any regular language with its DFA in linear time. For any regular language L, show that you can construct a family of Boolean circuits deciding L of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$.

4. **Cellular automata vs. Turing machines** [**10 marks**]
   Let's test the strong Church-Turing thesis by considering a new model of computation, cellular automata (CA), and showing that this model is polynomially equivalent to Turing machines.[1]

   Definition 1. A *one-dimensional cellular automaton* is a triple $\mathcal{A} = (\Gamma, b, \phi)$ where
   - a finite set of states, Γ,
   - a *blank state* $b \in \Gamma$,
   - an *update rule* $\phi\colon \Gamma^3 \to \Gamma$,

   and subject to the condition $\phi(b, b, b) = b$. A *configuration* is a function $c\colon \mathbb{Z} \to \Gamma$ such that $c(n) = b$ for all but finitely many values of $n \in \mathbb{Z}$.

   We start a computation by initializing consecutive cells with the input string (much like the Turing machine, but without a tape head to worry about). Then the computation

[1] If you have seen cellular automata before, you may have encountered definitions which were, e.g., limited to binary states, fixed to two dimensions, or where cells are updated based on counts of their neighbours. The definition here is not like these examples, so read it carefully.

proceeds in discrete steps $c_0, c_1, c_2, \ldots$ where the configuration updates deterministically as follows:

$$c_{t+1}(n) := \phi(c_t(n-1), c_t(n), c_t(n+1)) \quad \text{for all } n \in \mathbb{Z}.$$

A *description* $\langle c \rangle$ of a configuration c is an interval $c(i)c(i+1)\cdots c(j)$ of the function which contains all non-b states occurring in c. For this question, we say the computation halts if it reaches a *stable configuration* that does not change from one step to the next.

(a) Argue that for any cellular automaton $\mathcal{A}$ as defined above, there is a Turing machine $M_{\mathcal{A}}$ which takes a configuration $\langle c_0 \rangle$ and natural number $t \in \mathbb{N}$ as input, and simulates the CA by writing $\langle c_t \rangle$ on the tape and halting. Moreover, show that $M_{\mathcal{A}}$ halts in time polynomial in the size of c_0 (number of non-blank cells).

(b) In the other direction, show the following. For any Turing machine M, there exists a CA $\mathcal{A}_M$ such that for any input $x \in \{\mathtt{0}, \mathtt{1}\}^*$, the CA halts on input x if and only if M accepts x, and moreover halts in $\mathcal{O}(t)$ steps if the TM halts in t steps.

(c) Design a cellular automaton which beats the single tape Turing machine by deciding

$$\mathsf{PALINDROME} = \{x \in \{\mathtt{0}, \mathtt{1}\}^* \mid x = x^R\},$$

in linear time. Specifically, if x is a palindrome then it should accept by halting in the all-blank configuration, and reject by halting in any other configuration. Justify the correctness and runtime of your CA.