1. Kolmogorov complexity. Given a string $x \in \{0, 1\}^n$, recall that $K(x)$ is the length of the shortest program $A$ (in some Turing-universal programming language $P$) such that $A(\ ) = x$. We proved in class that the function $K$ is uncomputable.

(a) Let $K_P$ be the version of $K$ where we consider programs written in programming language $P$. Explain why, given two Turing-universal programming languages $P, Q$, there exists a constant $c_{PQ}$, depending only on $P$ and $Q$, such that $K_P(x) \leq K_Q(x) + c_{PQ}$ for all $x$. This is what justifies dropping the subscripts $P$ and $Q$, and simply writing $K(x)$. [Note: For this question, you don’t need to tie yourself in knots over the exact definition of “programming language.” It suffices to give an argument that would work for any programming language in ordinary use.]

(b) Show that $K \leq_T HALT$. (In fact $HALT \leq_T K$ as well! But that’s a harder result that we’re not asking you to prove.)

2. The Superhalt Turing-degree. Recall that $SUPERHALT = \{(\langle M \rangle, x) : M^{HALT}(x) \text{ halts}\}$.

(a) Let $L = \{\langle P \rangle : P(y) \text{ runs forever for some input } y \in \{0, 1\}^*\}$. Show that $L \leq_T SUPERHALT$.

(b) [Extra credit] Show that $SUPERHALT \leq_T L$. [Hint: Given $\langle M \rangle$ and $x$, you want to decide whether $M^{HALT}(x) \text{ halts}$, using an $L$ oracle. Construct a new Turing machine $P$, which doesn’t require an oracle, such that $P(k) \text{ runs forever for all large enough positive integers } k$, if and only if $M^{HALT}(x) \text{ halts}$.]

3. Fun with Gödel and Busy Beaver. Let $F$ be some system of axioms. You can assume $F$ is sound (that is, it only proves true statements), and also that $F$ is strong enough for Gödel’s Incompleteness Theorem to apply to it. Let $G(F)$ be the Gödel sentence of $F$ (that is, a mathematical encoding of “This sentence is not provable in $F$.”) Also, let $M_F$ be a Turing machine that generates all possible $F$-proofs, one by one, and that halts if and only if it encounters a proof of $G(F)$.

(a) Does $M_F$ halt? Why or why not?

(b) Is there a proof in $F$ that $M_F$ halts, or a proof in $F$ that $M_F$ that doesn’t halt? Why or why not?

(c) Recall that $BB(n)$, the $n^{th}$ Busy Beaver number, is defined to be the largest finite number of steps used by any $n$-state Turing machine, when run on an initially blank tape. Suppose our “Gödelian machine” $M_F$ has $k_F$ states. Show that for all $n \geq k_F$ and all $c$, the statement “$BB(n) = c$” cannot be proved in $F$—even if the statement is true! In other words: as long we restrict ourselves to $F$, there’s some absolute limit $k_F$ on the number of values of the Busy Beaver function that we can ever know. [Hint: Suppose otherwise; then use part b to derive a contradiction.]

(d) [Extra credit] Show that $BB(2) \geq 6$, for Turing machines over the alphabet $\{0, 1\}$, with a two-way-infinite tape initialized to all 0’s. The “HALT” state does not need to be counted as a separate state.
4. **The length of proofs.** Consider a formal system $F$, such as ZFC: that is, a formal language and a set of axioms and inference rules, for which there exists a computer program to check whether a given logical deduction is valid—and, as a consequence, another computer program to list all the possible valid deductions (that is, all the theorems) of $F$. You can assume that $F$ is powerful enough to express statements like “$M(\ )$ halts” or “$M(\ )$ runs forever,” where $M$ is a Turing machine; and that it’s easy to construct these statements given a description of $M$. You can also assume that $F$ is sound regarding such statements: in other words, if $F$ proves the statement “$M(\ )$ halts,” then $M(\ )$ halts, while if $F$ proves the statement “$M(\ )$ runs forever,” then $M(\ )$ runs forever. Finally, you can assume that whenever $M(\ )$ halts, $F$ proves the statement “$M(\ )$ halts” (for example, by just explicitly listing the steps until $M(\ )$ has halted).

Now, given a statement $S$ that’s provable in $F$, let $\ell_F(S)$ be the number of bits in the shortest $F$-proof of $S$, if we encode proofs as bit-strings in some canonical way. Also, let $\ell_F(n)$ be the maximum of $\ell_F(S)$, over all statements $S$ that can be encoded using at most $n$ bits.

Prove that, just like the Busy Beaver function, the function $\ell_F(n)$ grows faster than every computable function $f$. Or informally: there must exist short statements that are provable in $F$, but only via “absurdly long” proofs—even worse than Fermat’s Last Theorem, which takes only one line to state but (as far as we know today) hundreds of pages to prove. This necessity for enormous proofs was a 1936 observation of Kurt Gödel. [Hint: To prove this, it suffices to prove the stronger result that given any oracle that provided upper bounds on $\ell_F(n)$, you could solve the halting problem, compute Busy Beaver numbers, or do something else previously shown to be uncomputable.]

5. **Asymptotics.**

(a) Sort the following upper bounds, in order of which bounds logically imply which other bounds: $O(2^n), 2^{O(n)}, 2^{o(n)}, 2^{n^{O(1)}}, 2^{n^n}, O(3^n), 3^{O(n)}, 3^{o(n)}, 3^{n^{O(1)}}, 3^{n^n}$. Group together bounds that all imply one another (i.e., are equivalent). Brief explanations are encouraged but not necessary. [Note: Recall that $f(n) = 2^{O(n)}$ if solving $f(n) = 2^{o(n)}$ for $g$ yields $g(n) = O(n)$, and likewise for the other notations. Also, be careful! It’s easy to make mistakes on this problem.]

(b) For what function $f$ is it the case that $t = O(n^{\log n})$ if and only if $n = \Omega(f(t))$?

6. **Infinity.** Say whether each of the following infinite sets is countable or uncountable, and give a brief explanation why.

(a) The set of finite sets of integers.
(b) The set of languages $L \subseteq \{0, 1\}^*$ in the complexity class $P$.
(c) The set of languages $L \subseteq \{0, 1\}^*$ not in the complexity class $P$.
(d) The set of languages to which $HALT$ is Turing-reducible.
(e) The set of languages Turing-reducible to $HALT$.

7. Recall that $\text{EXP} = \bigcup_k \text{TIME} \left(2^{n^k}\right)$, and that $\text{NEXP} = \bigcup_k \text{NTIME} \left(2^{n^k}\right)$. Just as it is open problem whether $P = \text{NP}$, it is also an open problem whether $\text{EXP} = \text{NEXP}$. Show that, nevertheless, these two problems are closely related: if $P = \text{NP}$, then $\text{EXP} = \text{NEXP}$ also. [Hint: Given a language $L \in \text{NEXP}$, can you come up with a modified language $L' \in \text{NP}$, such that $L \in \text{EXP}$ if and only if $L' \in \text{P}$?]

8. Show that $\text{PSPACE} \neq \text{TIME} \left(2^n\right)$. [Hint: Suppose by way of contradiction that $\text{PSPACE} = \text{TIME} \left(2^n\right)$. Then can you use another padding argument to violate the Time Hierarchy Theorem?]

2