Problem 1. Let $L$ be a language. Prove that $L \in DEC$ if and only if there is an enumerator that prints $L$ in standard order (i.e. the computable/enumerable sequence $\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$).

Problem 2. Let $A, B \in RE$. Give a Turing machine (with no proof necessary, though an explanations are appreciated) that shows

a) $A \cap B \in RE$.

b) $A \cup B \in RE$.

c) $AB \in RE$.

d) $A^* \in RE$.

Notably, $RE$ is not closed under complementation (though $DEC$ is).

Problem 3. An $n$ busy beaver is a Turing machine of $n$ states that, with a two-way infinite blank tape (and thus no input) and tape alphabet $\Gamma = \{\bot, 1\}$, first prints as many 1’s onto the tape as possible and then halts. A busy beaver is a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ that is a $|Q|$ busy beaver.

The busy beaver problem

$$BB = \{\langle M \rangle \mid M \text{ is a busy beaver}\}$$

is well-known to be undecidable. Explain briefly why (you do not need to give a formal proof) this must be the case.

Problem 4. Prove that the following language $L$ is undecidable by (Turing) reducing $BB$ to it.

$$L = \{\langle M, i \rangle \mid M(\epsilon) \text{ halts with at least } i \text{ non-blank symbols on its tape}\}$$

(Hint: Compute the number of 1’s an $|M, Q|$ busy beaver produces first, then determine if $M$ is a busy beaver; you will need to make multiple requests of your oracle for $L$)

Problem 5. Recall that for a function $f$ to be computable, there must be a Turing machine which computes it, meaning on every input $\omega$, $M$ halts with $f(\omega)$ on its tape and nothing else. This applies to functions whose inputs and outputs are natural numbers as well since there is a (computable) bijection between $\Sigma^*$ and $\mathbb{N}$. For example, you have
previously shown that \( f(x) = x + 1 \) and \( f(x) = 2x \) are computable. This allows us to state some theorems more naturally.

The set of all computable functions \( f : \mathbb{N} \to \mathbb{N} \) is countable. However, the set of all functions \( f : \mathbb{N} \to \mathbb{N} \) is uncountable. Consequentially, almost all such functions are not computable. This is the same argument we made to show that almost all languages are not in \( \text{RE} \) or \( \text{co-RE} \). Beyond this, there are many other strange properties of uncomputable functions that may surprise you.

Prove that there exists a function which grows faster than any computable function. In other words, show that there is a function \( f : \mathbb{N} \to \mathbb{N} \) such that for any computable function \( g : \mathbb{N} \to \mathbb{N} \) there exists an \( N \in \mathbb{N} \) such that for all \( n \geq N \), \( f(n) > g(n) \).

More succinctly, show \( \exists f : \mathbb{N} \to \mathbb{N} : \forall g : \mathbb{N} \to \mathbb{N} \text{ computable} \forall n, f(n) > g(n) \).

(Hint: Approach this similarly to dovetailing by overcoming one additional computable function at a time)