Problem 1. (Universal Turing Machines) [0 points] (Discarded) Let \( \Sigma \) be an alphabet with \( |\Sigma| \geq 2 \) and pick two distinct characters \( a, b \in \Sigma \). We could encode a natural number \( n \) as the string \( a^n b \), but there is a more compact way. We write \( n \) in binary (using \( a \) for zero and \( b \) for one) with each digit prefixed by an \( a \) and with a \( b \) at the end. For instance, we encode \( 0, 2 \) and \( 10 \) by \( b \), \( abaab \) and \( abaaabaab \). More precisely, a natural number \( n \) is encoded as the string 
\[
\langle n \rangle = \left( \prod_{i=\lceil \log_2(n+1) \rceil - 1}^{0} ax_i \right) b
\]
where \( x_i = a \) if \( \lfloor n/2^i \rfloor \) is even and \( x_i = b \) otherwise.

Let now \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R) \) be a Turing Machine with the same \( \Sigma \) as above. We can assume, without loss of generality, that \( Q = \{q_0, \ldots, q_n, q_A = q_{n+1}, q_R = q_{n+2}\} \) with \( n \in \mathbb{N} \) and \( \Gamma = \{\gamma_0, \ldots, \gamma_{m-1}\} \) with \( m \in \mathbb{N} \) and \( m \geq 2 \) (recall \( \Sigma \subseteq \Gamma \)). We wish to encode \( M \) as a string in \( \{a, b\}^* \) and we begin by encoding the head movement directions: \( \langle \leftarrow \rangle = a \) and \( \langle \rightarrow \rangle = b \). Next we encode a triple \((q, \gamma, d) \in Q \times \Gamma \times \{\leftarrow, \rightarrow\}\) as the string \( \langle (q, \gamma, d) \rangle = \langle i \rangle \langle j \rangle \langle d \rangle \), where \( q = q_i \) and \( \gamma = \gamma_j \). This is so we can encode the transition function \( \delta : (Q \setminus \{q_A, q_R\}) \times (\Gamma \cup \{\_\}) \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\} \) as the string 
\[
\langle \delta \rangle = \prod_{i=0}^{n} \left( \langle \delta(q_i, \_\) \rangle \prod_{j=0}^{m-1} \langle \delta(q_i, \gamma_j) \rangle \right).
\]

Finally, we simply encode \( M \) as the string \( \langle n \rangle \langle m \rangle \langle \delta \rangle \).

A Universal Turing Machine is a Turing Machine \( U \) that checks its input is of the form \( \langle M \rangle w \) where \( M \) is a Turing Machine and \( w \in \Sigma^* \) (rejecting otherwise) and then “emulates” \( M \) on input \( w \). More precisely, it accepts if \( M \) accepts \( w \), rejects if \( M \) rejects \( w \) and diverges if \( M \) diverges on \( w \).

In this problem, you are asked to construct such Universal Turing Machine. Recall that we have (more or less) established that programming languages such as Python are not more expressive than Turing Machines. Therefore, it is enough to write a program in such a programming language that: reads an input; checks it has the form \( \langle M \rangle w \); and “emulates” \( M \) on \( w \), returning, say, True if \( M \) accepts \( w \) and False if \( M \) rejects \( w \), and never returning, i.e., computing forever, if \( M \) diverges on \( w \). You do not need to show code in your solution, but you need to: briefly describe how you parse the input, describe which data structures your program uses, state its loop invariant and state its loop termination condition.
Problem 2. \textbf{[25 points]} Prove that a language $A$ over an alphabet $\Sigma$ is decidable if, and only if, $A$ and $\Sigma^* \setminus A$ are enumerable. If you construct a Turing Machine, a broad overview of how it operates is enough. Hint: think about how you could adapt the machine $U$ from Problem 1.
Problem 3. (Regular operations on decidable and enumerable languages) [60 points] Prove that the following statements are all true. A high-level description of any Turing Machine you employ is enough, but make sure you show you understand what they do.

- The union of two decidable languages is decidable;
- The union of two enumerable languages is enumerable;
- The concatenation of two decidable languages is decidable;
- The concatenation of two enumerable languages is enumerable;
- If $A$ is a decidable language, then so is $A^*$; and
- If $A$ is an enumerable language, then so is $A^*$.  

Problem 4. [15 points] Suppose we wish to show that all context-free languages are decidable by implementing the CYK algorithm on the $\lambda$-calculus. An implementation of matrices might then make itself useful. Show a correct implementation of matrices on the $\lambda$-calculus, i.e., define the following $\lambda$-terms (the precise specification is omitted, but your answer must make sense):

- **MakeMatrix**: takes as arguments the number of rows and columns as Church numerals and an initial value; returns a matrix with that many rows and columns and with each cell initialized to the given value.

- **MatrixGet**: takes as arguments a matrix and the (0-based) row and column indices of a cell as Church numerals; if the row and column indices are within range, returns the value at the indexed cell; if the indices are out of range, you are allowed to have it perform however you like (undefined behavior).

- **MatrixSet**: takes as arguments a matrix, the (0-based) row and column indices of a cell as Church numerals and a new value; if the row and column indices are within range, returns a matrix that equals the input one on every cell but the indexed one, where it has the new value; if the indices are out of range, you are allowed to have it perform however you like.