1. Concept Check: $\mathrm{U_{TM}}$, $\mathrm{A_{TM}}$, Verifiers, L_D

a. What is the difference between $\mathrm{A_{TM}}$ and the $\mathrm{U_{TM}}$?

b. The $\mathrm{U_{TM}}$'s inputs have the form $<M, w>$. What are M and w? What do the angle brackets represent? How does the $\mathrm{U_{TM}}$ behave on this input?

c. A **verifier** V for a language L is a TM with these two properties:

$$V \text{ halts on all inputs, and } \forall w \in \Sigma^*.\ (w \in L \leftrightarrow \exists c \in \Sigma^*.\ (V \text{ accepts } <w, c>))$$

Explain in your own words what this definition means. Then, explain how you would formally prove that a TM is a verifier.

d. Which one of these statements is true?

(1) A verifier for a language L is a decider for L.

(2) A verifier for a language L is a decider for some other language, not L.

(3) A verifier for a language L is not a decider for any language.

e. L_D is defined as $\{<M> \mid \text{M is a TM and M does not accept } <\text{M}>\}$. Explain this definition.

2. Self-Reference and Undecidability

We'll show the language $L = \{<M> \mid M \text{ is a TM that accepts the string } \texttt{"CS"}\}$ is undecidable.

a. Your friend says, "L is the language of all TMs that accept the string `"CS"`. I can make a TM M that accepts the string `"CS"`. M is in L. M is a decider for L." Explain which part of your friend's statement is wrong.

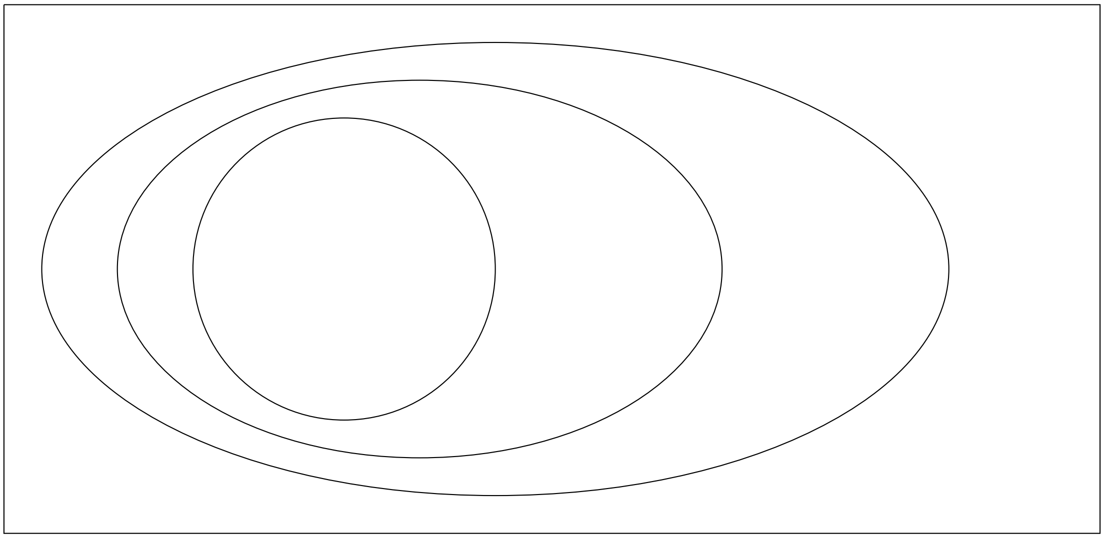
b. Suppose for the sake of contradiction that $L \in \mathbf{R}$. This means that we could write a function `bool acceptsTheStringCS(string program)` that takes source code as input, then returns true if the program accepts the string "CS" and returns false otherwise. Fill in the following self-referential program that uses this function to obtain a contradiction.

```
bool trickster(string w) {
    string me = /* source code of trickster */;

    if (acceptsTheStringCS(me)) {
        // In this case, the decider says that this program will accept the
        // string "CS". The program should do the opposite of that:
        ________________________
    } else {
        // In this case, the decider says that this program will NOT accept
        // the string "CS". The program should do the opposite of that:
        ________________________
    }
}
```

c. Explain why the program you just wrote leads to a contradiction.

3. Placing Languages in the Lava Diagram



a. Fill in the blanks and place the labels in the right spots on the diagram.

(1) All languages

(2) **RE** (languages that have a ________________ or ________________)

(3) **R** (languages that have a ________________)

(4) **REG** (languages that have a ____________, ____________, or ____________)

b. Categorize these languages on the diagram:

(1) $\{a^n b^n \mid n \in \mathbb{N}\}$

(2) $\{<M> \mid M$ is a TM that accepts the string "cool"$\}$

(3) $\{<M> \mid M$ is a TM that accepts ONLY the string "cool"$\}$

(4) $\{w \in \Sigma^* \mid |w| \leq 103\}$

(5) $\{<M> \mid M$ loops on all inputs$\}$

(6) $\{a^{2n} \mid n \in \mathbb{N}\}$

(7) $\{a^{2^n} \mid n \in \mathbb{N}\}$

4. Exploring Verifiers

Consider the language $L = \{\texttt{a}^n\texttt{b}^n \mid n \text{ is a multiple of } 103\}$.

a. Each of the following TMs is a verifier for L. Explain why.

   (1) V is a decider for the language $\{<w, n> \mid w = \texttt{a}^k\texttt{b}^k \land k = 103n\}$

   (2) V is a decider for the language $\{<w, \varepsilon> \mid w \in L\}$

b. Each of the following TMs is **not** a verifier for L. Explain why not.

   (1) V accepts strings of the form $\texttt{a}^n\texttt{b}^n$, where n is a multiple of 103, and rejects other strings

   (2) V is U_{TM}, the universal Turing Machine

   (3) V is a decider for the language $\{<w, n> \mid n \text{ is a multiple of } 103\}$.

   (4) V is a decider for the language $\{<w, n> \mid w = \texttt{a}^n\texttt{b}^n\}$