Due April 30, 4:35 p.m.

We've covered a lot of terminology and concepts pertaining to Turing machines and R and RE languages. These problems explore some of the nuances of how Turing machines, languages, and decidability relate to one another. By working through them you should gain a better understanding of some key concepts in computability.

**Problem 1**

What does it mean to solve a problem? If $L$ is a language over $\Sigma$ and $M$ is a Turing machine with input alphabet $\Sigma$, any of these properties may hold:

1. $M$ is a decider (i.e., halts on all inputs).
2. For any string $w \in \Sigma^*$, if $M$ accepts $w$, then $w \in L$.
3. For any string $w \in \Sigma^*$, if $M$ rejects $w$, then $w \notin L$.

To claim that a Turing machine solves a problem, it seems like it should have at least some of these properties. But we can show that having just two of these properties doesn't say much.

a. Prove that if $L$ is any language over $\Sigma$, then there is a Turing machine $M$ that satisfies properties 1 and 2 with respect to $L$.

b. Prove that if $L$ is any language over $\Sigma$, then there is a Turing machine $M$ that satisfies properties 1 and 3 with respect to $L$.

c. Prove that if $L$ is any language over $\Sigma$, then there is a Turing machine $M$ that satisfies properties 2 and 3 with respect to $L$.

d. Suppose $L$ is a language over $\Sigma$ for which there is a Turing machine $M$ that satisfies properties 1, 2, and 3. What can you say about $L$?

**Problem 2**

We've introduced several terms and definitions related to Turing machines, languages, and what it means to solve a problem. Some of these terms can only describe Turing machines, while others can only describe languages. A statement that uses terms incorrectly has no meaning; it cannot be evaluated as being true or false.

Consider, by analogy, the statement “the set $\mathbb{N}$ is even”. This statement doesn't type-check because “even” only applies to individual numbers, and $\mathbb{N}$ isn't a number. Similarly, the statement “$1 \in 5$” doesn't type-check because 5 isn't a set. On the other hand, the statement $\mathbb{Z} \subseteq \mathbb{N}$ type-checks but it isn't true – the integers aren't a subset of the natural numbers. This is the mathematical equivalent of a grammatically correct sentence that's false, like “Otters aren't cute”. You understand what the sentence means, but it's objectively not true. They're adorable.
Below is a series of statements. For each, decide whether

- the statement type-checks and is true,
- the statement type-checks and is false, or
- the statement does not type-check.

Briefly explain your answers.

a. If $M$ is a Turing machine, $w$ is a string, and $M$ accepts $w$, then $A_{TM}$ accepts $\langle M, w \rangle$.

b. If $M$ is a Turing machine, $w$ is a string, and $M$ loops on $w$, then $\langle M, w \rangle \notin L(U)$.

c. $U$ is decidable.

d. $\langle U \rangle$ is decidable.

e. $\{\langle U \rangle \}$ is decidable.

**Problem 3**

Give a high-level description of a Turing machine $M$ such that $L(M) \in \mathbb{R}$, but $M$ is not a decider. (This shows that just because a Turing machine's language is decidable, it's not necessarily the case that the TM itself must be a decider.)

**Problem 4**

In class, we saw that $A_{TM}$ is undecidable but Turing-recognizable. From a programming perspective, this means that it's possible to write a procedure `will_accept(program, input)` that takes as its arguments string representations of a program and that program's input and has the following return values:

- If the program would accept that input, `will_accept` *must* return true.
- If the program would not accept the input, `will_accept` *may* return false or it may go into an infinite loop.

Now consider the following program, `ow-my-brain.py`:

```python
def will_accept(program, input):
    # An implementation of a recognizer for ATM: Does the program accept the
    # input?
    ...

def main(my_input):
    # Read our own source code
    my_code = open("ow-my-brain.py").read()

    if will_accept(my_code, my_input):
        return False # Reject
    else:
        return True # Accept
```
Use proof by contradiction to show that this program must loop infinitely on all inputs.

**Problem 5**

Show that the collection of decidable languages (R) is closed under the intersection operation.

**Problem 6**

Prove or disprove: The following language is decidable:

\[ L = \{ (M, n) \mid M \text{ is a Turing machine with } n \text{ states, and there exists a string } w \text{ in } \Sigma^* \text{ of length at most } n^2 \text{ such that } M \text{ accepts } w \text{ in at most } n^3 \text{ transitions } \} \]

Follow the style of the solutions to similar problems in the textbook, e.g., 4.10, 4.12, and 4.14 on pages 213–4.