Reminder: Your name should not appear anywhere on your handin; each individual page of the homework should include your Banner ID only. For your digital submission, each page should include work for only one problem (i.e., make a new page/new pages for each problem).

While collaboration is encouraged in this class, please remember not to take away notes from any labs or collaboration sessions, and to list your collaborators at the beginning of your homework submission.

Please monitor Piazza, as we will post clarifications of questions there. Hand in your solutions by 10:20am to Gradescope. See the course syllabus for the late policy.

Problem 1

Explain why the following is not a description of a legitimate Turing Machine.

\[ M_{bad} = \text{"The input is a polynomial } p \text{ over variables } x_1, ..., x_k. \]

1. Try all possible settings of \( x_1, ..., x_k \) to integer values
2. Evaluate \( p \) on all of these settings.
3. If any of these settings evaluates to 0, accept; otherwise reject.

Solution: All possible settings of \( x_i \) are infinite, as there are infinite integers. Therefore, step 2 will never yield to step 3, so if a reject state *should* occur (i.e. there are no settings of \( x_i \) that make \( p \) evaluate to 0), we would never be able to reach that reject state, and if an accept state *should* be reached (i.e. we have found a suitable setting), we still have to iterate through infinite settings before being allowed to reach that state. Looping on its own is obviously not a problem, (you do not need to halt on all inputs to be a Turing-Recognizable language), however this machine puts its accept and reject states after a never ending search through infinite combinations of integers, so it won’t halt on any input at all. So it’s not a legitimate Turing Machine because it does not have a defined reject state
or accept state – they literally *never* can be reached and the machine will always loop forever.

**Problem 2**

A *Turing machine with left reset* is similar to an ordinary Turing machine, but the transition function has the form:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, \text{RESET}\}$$

If $\delta(q, a) = (r, b, \text{RESET})$, when the machine is in state $q$ reading an $a$, the machine’s head jumps to the left-hand end of the tape after it writes $b$ on the tape and enters state $r$. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

**Solution:** We can simulate a regular Turing Machine using the TM with left reset in the following way: When the regular TM would move right, it moves right; and when the regular TM would move left, it marks its current space, resets to the left, copies every space one space to the right (maintaining the position of the mark), then resets and moves to the position of the mark. The copying can be done in a single sweep, by simultaneously remembering the symbol that is being copied and the symbol that is being overwritten.

Bonus: we can also simulate the left-reset TM in the regular TM. When the left-reset TM would move right, regular TM moves right; and when the left-reset TM would reset to the left, the regular TM moves left repeatedly not changing any letters it sees until it gets to the left end of the tape.

**Problem 3**

Show that the collection of decidable languages is closed under the operation of

1. concatenation
2. intersection
3. star
Solution:

1. Suppose we have any two decidable languages in the collection of decidable languages, A and B. Because they are decidable, there are Turing Machines that decide them, let's call them $TM_A$ and $TM_B$. To show that their concatenation is also decidable, let's construct $T_{concat}$ that decides by simulating $TM_A$ followed by $TM_B$ on an input string $w \in \{w_Aw_B|w_A \in A, w_B \in B\}$. Since we don't know how an input string will be broken up into $w_A$ and $w_B$, we will try all possible splits (which will require looping over the length of the string). At the start, we can write $w$ on the tape with a special symbol marking the split at the end of the input string, and move the position of this special symbol one to the left for each iteration. Then, we run $TM_A$ on everything before the special symbol and if it accepts, run $TM_B$ on everything after and if it accepts, accept the string, otherwise, try again with the symbol moved one to the left. If none of the splits accept, reject the string. Since we can construct a TM that can decide the concatenation of any decidable languages, we have proven that decidable languages are closed under concatenation.

2. To show that decidable languages are closed under intersection, we must show that given any two languages, their intersection is decidable by some Turing Machine $M'$. Suppose we have decidable languages $A$ and $B$ that are decided by turing machines $TM_A$ and $TM_B$. Let $M'$ make decisions about input string $w$ in the following way: run $TM_A$ on $w$; if rejected, reject $w$; if accepted, run $TM_B$ on $w$; if accepted, accept $w$, otherwise reject. What this does is guarantee that $w$ belongs to both language $A$ and $B$, thus belonging to language $L(M') = A \cap B$, making $A \cap B$ decidable. Therefore, decidable languages are closed under intersection.

3. Suppose we have some decidable language $A$ that is decided by $TM_A$. We now want to construct a Turing machine $M'$ that decides $\{w^*|w \in A\}$ as follows: if the input string $s$ is empty, accept. Otherwise, run $TM_A$ on the first character of $s$, which we will call $s'$; if $TM_A$ accepts, see if the rest of the string consists of $s'$ repeated until the end of the string, and if so accept; otherwise, extend $s'$ to include the subsequent character and repeat the process above. If $s$ is not in $A$ or $s$ is not composed of some number of full repetitions of $s'$ where $s'$ is in $A$, then $s$ will be rejected by $M'$, otherwise, it will be accepted. Since
we have constructed a TM that can decide the language of any decidable language with the star operation, we can conclude that decidable languages are closed under star.