Each problem set counts for 10% of your mark. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Copying assignments is a serious academic offense and will be dealt with accordingly.

You may use any results discussed in lecture, the course textbook, or tutorial. It should be possible to write your solutions to the problem set within 2-3 pages of work. Submit your work online by uploading a pdf file or image of your solutions onto Crowdmark.

1. Let $F$ be the language $F = \{a^ib^jc^k : i + j = k, i, j, k \geq 1\}$. Show that $F$ is not regular and furthermore, describe (in English or pseudocode) a Turing machine that accepts $F$.

**Solution:** We will use the pumping lemma to show that $F$ is not regular. If $F$ is regular, then there is some pumping length $p \geq 1$ where all strings of length in $F$ with length greater than $p$ satisfy the conditions of the pumping lemma. Choose $w = a^pb^pc^{2p} \in F$ and note that any decomposition of $w$ into substrings $xyz$ with $|y| \geq 1$ and $|xy| \leq p$ means that $y = a^i$ for some $1 \leq i \leq p$. Then if $F$ is regular, then $xz = a^{p-i}b^pc^{2p} \in F$. But $(p - i) + p = 2p - i \neq 2p$ so $xz \notin F$ and so $F$ cannot be regular.

We will now describe a Turing machine that accepts inputs in $F$. The Turing machine has the following description.

(1.) Scans its tape from left to right, checking that the input consists of $a$s followed by $b$s, then followed by $c$s. If not, the machine rejects.

(2.) Reset the tape head and find the leftmost unmarked cell. If the tape head reads $a$ or $b$, mark it by replacing it with ˚$a$ or ˚$b$ respectively, and scan the tape head to the right to find the leftmost unmarked $c$. If unmarked $c$ does not exist, reject, and otherwise mark the $c$ by replacing it with ˚$c$.

(3.) Repeat the previous step until either all characters in the input are marked, in this case we accept, or otherwise we find an unmarked $c$, in this case we reject.

The Turing machine accepts $F$ since Step 1 will reject all inputs not of the form $a^ib^jc^k$ for some $i, j, k \geq 1$, Step 2 will reject all inputs $a^ib^jc^k$ with $i + j > k$, and Step 3 will reject all inputs $a^ib^jc^k$ with $i + j < k$. Therefore, only inputs of the form $a^ib^jc^k$ with $i + j = k$ are accepted.

2. A **Turing machine with left reset** is a Turing machine with its transition function replaced by

$$
\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{R, \text{RESET}\}.
$$
If $\delta(q,a) = (q', b, \text{RESET})$, then the Turing machine writes $b$ at its current tape position, and afterwards, the tape head moves to the leftmost position on the tape and the machine goes into state $q'$.

Show that the set of languages recognized by Turing machines with left reset is the same as the set of languages recognized by Turing machines with left and right moves only.

**Solution:** It is clear that a Turing machine $M'$ left reset can be simulated by a Turing machine $M$ with right and left moves only, so it suffices to show that a left move can be simulated by a combination of right moves and left reset to prove the claim. We will describe $M$ can be simulated by $M'$ (except possibly using a larger tape alphabet) in the following way.

Suppose $M$ wants to perform a transition $\delta(q,a) = (q', b, L)$. Then $M'$ simulates $M$ in state $q$ when reading $a$ on the tape by firstly writing a marked symbol $\dot{b}$ on the current position, moving right to mark the end of the non-empty part of the tape with $\#$, and resetting. Next, $M'$ uses new states to shift its current contents on its tape one cell to the right and remember the next state $q'$ it should transition to. Furthermore, if $\dot{b}$ is read, and $M'$ is in a state that remembers the cell to the left of it contained $x$ originally, it replaces $\dot{b}$ with $\dot{x}$. Furthermore, when $\dot{b}$ is shifted to its immediate right, it is unmarked. Once the $\#$ is replaced, $M'$ resets and finds the marked symbol $\dot{x}$ by scanning the tape from left to right. Once $\dot{x}$ is found, the tape head of $M'$ is now reading a symbol that is one cell left of its original position and $M'$ remembers that it transitioned to state $q'$ before the reset. Hence $M'$ has performed a left move with right and reset only.

Other solutions are possible. Another possible solution involves after marking the current position, resetting, and doing a linear search over the tape to find the position one cell left using two marked cells.

3. Prove that a language $A \subseteq \Sigma^*$ is semi-decidable if and only if there is a decidable binary relation $R \subseteq \Sigma^* \times \Sigma^*$ such that for all $x \in \Sigma^*$, $x \in A$ if and only if there is some $y \in \Sigma^*$ for which $(x, y) \in R$.

A binary relation $R$ is **decidable** if there is a Turing machine $M$ where if a pair $(x, y) \in \Sigma^* \times \Sigma^*$ is given as input, then $M$ determines whether or not $(x, y) \in R$ in finite time.

**Solution:** If a decidable relation $R$ exists, then a Turing machine $M$ can on input $x$ test whether or not $(x, y) \in R$ is true for $y \in \Sigma^*$ listed in lexicographic order, and accepting once $y$ is found, if it exists. This Turing machine $M$ by definition of $R$ accepts only the strings $x \in A$.

Conversely, if $A \subseteq \Sigma^*$ is semi-decidable, there is a Turing machine $M$ recognizing $A$ and we have to construct a decidable binary relation $R$ such that $x \in A$ if and only if there is some $y$ with $(x, y) \in R$. Many choices of $R$ are possible. One choice of relation $R$ is the set $(x, y)$ where $y$ is the number of steps $M$ takes to accept on input $x$. This number $y$ can be encoded according to the alphabet $\Sigma$ in unary (if $a \in \Sigma$ is a symbol, the number $n$ is represented by the string $a^n \in \Sigma^*$.) Furthermore, the relation $R$ is decidable since a Turing machine can determine whether or not $(x, y) \in R$ by running $M$ for at most $y$ steps, and then accepting if $M$ has accepted $x$ within that time (and rejecting otherwise).

4. Show that every infinite semi-decidable language has an infinite decidable subset.

**Solution:** Recall that if $A$ is a semi-decidable language, then there is an enumerator $E$ that prints the strings of $A$ in some order. List the strings in $A$ according to that order.
\{x_0, x_1, x_2, \ldots \}$ printed by the enumerator. We inductively define a subset $B \subseteq A$ by declaring that $x_0 \in B$, and for $j > 0$, $x_j \in B$ if and only if its length is less than all those strings $x_i$ with $i < j$. By construction, $B \subseteq A$ and it is infinite, since $A$ must contain strings of arbitrarily large length if $A$ was infinite.

Now we must argue that the set $B$ is decidable. This is because we can modify the enumerator $E$ to construct an enumerator $E'$ that prints $B$ in lexicographic order. We may assume that $E'$ has two work tapes and one printing tape by the theorem that multi-tape Turing machines can be simulated by single-tape Turing machines. The enumerator $E'$ has following description:

1. Simulate $E$ on the working tapes until it prints the first string $x_0$. Copy $x_0$ onto the printing tape of $E'$.
2. Continue simulating $E$ on the working tapes until it prints a new string $x_i$.
3. Copy $x_i$ on the printing tape of $E'$ if and only if it is longer than the previous string printed it.
4. Continue the simulation by going back to step 2.

Since the printing tape of $E'$ by definition prints the subset $B$ and furthermore does the printing in lexicographic order, we can conclude that $B$ is decidable, using the theorem that a language is decidable if and only if an enumerator prints it in lexicographic order.