PART B (Graded by Joy and Anupa)

Note: You may use implementation-level descriptions (Lecture 14, slide 14) for this part of the problem set unless the problem specifies otherwise.

PROBLEM 1 (10 points)

Define $\text{Prefix}(L) = \{x \mid xy \in L \text{ for some } y \in \Sigma^*\}$. Show that if $L$ is Turing-recognizable, then $\text{Prefix}(L)$ is Turing-recognizable.

Solution 1: We use a recognizer $M$ for $L$ to construct a recognizer $R$ for $\text{Prefix}(L)$: On input $x$, $R$ will dovetail the computation of $M$ on all strings $xy, y \in \Sigma^*$, accepting if $M$ accepts any $xy$. This will accept if and only if $x$ is a prefix of some string in $L$, so $R$ recognizes $\text{Prefix}(L)$.

Solution 2: Since $L$ is recognizable, there exists an enumerator $E$ for $L$. We construct a recognizer $R$ for $\text{Prefix}(L)$ as follows: On input $x$, $R$ runs $E$, and for every string $w$ $E$ outputs, accepts if $x$ is a prefix of $w$. This will accept if and only if $x$ is a prefix of some string in $L$, so $R$ recognizes $\text{Prefix}(L)$.

Solution 3: We construct an enumerator $R$ for $\text{Prefix}(L)$ using an enumerator $E$ for $L$: $R$: run $E$. For every string $w$ output by $E$, output all prefixes of $w$. This will enumerate exactly the strings in $\text{Prefix}(L)$, which proves that $\text{Prefix}(L)$ is recognizable.

PROBLEM 2 (5 points)

A Modern Turing Machine (MTM), instead of $\{L, R\}$, has $\{L_k, R_k\}$ (move the head left or right $k$ spaces on the tape, for any integer $k$ encoded in the rule in the MTM). Prove or disprove: the MTMs are equivalent in power to the TMs.
True. Every TM is already an MTM in which \( k = 1 \) for every rule. To convert an MTM to an equivalent TM, replace every \( k \)-transition (where \( k > 0 \)) with a set of \( k \) single transitions in the same direction, where all but the first read any tape symbol and do not alter it. Replace any 0-transitions with one right move (altering the letter as described in the 0-transition) and one left move (reading the symbol but not altering it). (It is important to move right, then left – not the other way around – since moving left may fail if we are at the left end of the tape.)

**PROBLEM 3 (6+6 points)**

A function \( f : \Sigma^* \to \Sigma^* \) is **computable** if there is a Turing machine \( M \) that, on every input \( w \in \Sigma^* \), halts (in state \( q_{\text{accept}} \)) with the tape containing just the string \( f(w) \) (followed by blank symbols). In this case, we say that \( M \) **computes** the function \( f \).

(A) Draw a state diagram for a (one-tape, deterministic) Turing machine \( M \) that performs unary addition. More specifically, it should take inputs of the form \( "a^ma^n" \), where \( m \) and \( n \) are positive integers, and output \( a^{m+n} \). Your solution should not require more than four or five states. (For example, running \( M \) on input \( aa\$a \) should result in \( aaa \) on the tape. You may assume that the input is well-formed.)

(B) Show that a function \( f \) over alphabet \( \Sigma \) is computable if and only if the language \( L = \{ w\$i\sigma : \text{the } i\text{'th symbol of } f(w) \text{ is } \sigma \} \) is decidable. (Thus, we can study computability of functions by studying decidability of languages.)

(A)

```
(a) $ \rightarrow a, R$
(b) \( a \rightarrow R \)
(c) \( a \rightarrow R \)
(d) \( L \rightarrow \_, L \)
(e) \( q_3 \rightarrow \_, R \)
(f) \( \text{accept} \)
```

(B) Assume first that \( f \) is computable, with corresponding Turing Machine \( M_f \). We will construct a Turing Machine \( M \) that decides \( L \). Our machine will use two tapes, crucially relying on the fact proved in class that a \( k \) tape Turing Machine is equivalent in power to a one tape Turing Machine. The first tape will be the input tape. The machine will work as follows:

1. Scan input \( s \) on tape 1. If \( s \) is not of the form \( w\$i\sigma \), for \( w \in \Sigma^*, \sigma \in \Sigma \), reject.

2. Copy the characters of \( s \) before the first \( \$ \) sign onto tape 2, i.e., copy \( w \) onto tape 2.

3. Simulate \( M_f \) on tape 2 with input \( w \).

4. Move the tape head on tape 1 to the first dollar sign and the tape head on tape 2 to the first character of \( f(w) \).

5. Advance both tape heads one symbol at a time to the right, crossing off characters until you reach \( \sigma \) on the first tape. If it matches the character under the tape head on tape 2, accept. Otherwise, reject.
This will reject any string not in $L$ either in step 1 if it is of the wrong form, or in step 5 if the $\sigma$ is not the $i^{th}$ character of $f(w)$. The machine will accept all strings in $L$ at step 5 because it checks for exactly the defining characteristics of a string in $L$.

Assume now that $L$ is decidable with decider $M$. We will construct a Turing Machine $M_f$ that computes $f$. This Turing Machine will have four tapes to minimize its conceptual difficulty. The first tape will be the input/output tape where $w$ is given and $f(w)$ is returned. It works as follows:

1. Copy $w$ onto the second tape.
2. Affix a single dollar sign to the end of the string on tape 2.
3. For each $\sigma \in \Sigma$, simulate on tape 3 the Turing Machine $M$ with $\sigma$ affixed to the end of the string on tape 2: i.e., compute $w\$\sigma$ on machine $M$.
4. If for any character $\sigma$, $M$ accepts the string, copy $\sigma$ to the rightmost empty tile on tape 4 and repeat steps 2 through 4. If $M$ rejects for all $\sigma \in \Sigma$, go to step 5
5. Copy the output of tape 4 to tape 1 and return. This is $f(w)$.

This machine works by taking $w$ and using $M$ to compute the characters of $f(w)$ on an individual basis. It simply checks all of the alphabet symbols to see which one is the $i^{th}$ character of $f(w)$. It halts when there is no $\sigma \in \Sigma$ such that $\sigma$ is the $i^{th}$ character of $f(w)$, thus indicating $f(w)$ has no $i^{th}$ character and thus that we must be at the end of $f(w)$. 