COMP 3721: Theory of Computation
assignment3

1. We assume initially that the tape is $\rhd\underline{\sqcup}\sigma_1\sigma_2\ldots\sigma_n\sqcup$. The Turing machine below performs the required task.:

$$
\begin{tikzcd}
> R \arrow[r, "\bar{\sqcup}"] \arrow[d, "\sqcup"] & R \arrow[r, "\sigma_1 \neq \sqcup"] \arrow[d, "\sqcup"] & \# R_{\sqcup} \arrow[r] & L \arrow[r, "\sigma_2 \neq \#"] \arrow[dl, "\#"] & R\sigma_2 L \arrow[l, bend left] \\
L & L_{\sqcup} & R\sigma_1 \arrow[ul] & &
\end{tikzcd}
$$

2. a) **Disprove:** Here is a counterexample. Let $L_1 = \Sigma^*$ and $L_2 = H$, where $H = \{"M","w" : \text{Turing machine } M \text{ halts on input } w\}$ known as the Halting language. Since H is undecidable, $L_1 - L_2 = \bar{H}$ is undecidable, and thus not regular.

   b) **Prove:** Since L_1 is r.e., there is a TM M_1 that **semi-decides** it. And since L_2 is context-free, there is a TM M_2 that **decides** it. We construct another TM M_0 behaving as follows: for any input string w, it first simulates M_2 on w, if $M_2(w)$ rejects, M_0 will loop forever; if $M_2(w)$ accepts, M_0 proceeds to simulate M_1 on w, and if and only if $M_1(w)$ halts, M_0 halts. That is, for any string w, M_0 will halt on w if and only if $w \in L_1 \cap L_2$. Thus, M_0 semi-decides $L_1 \cap L_2$, that is, $L_1 \cap L_2$ is r.e.

3. " $\Leftarrow$ " Suppose there is a Turing machine M that enumerates the strings of L in a non-decreasing order of their lengths. Then we can construct a Turing machine M' that decides L. The Turing machine M' performs as follows: for any input w, it simulates M until M has just enumerated w or has enumerated all the strings of L of lengths no longer than $|w|$. If M has just enumerated w, M' accepts w; if M has finished enumerated all the strings of lengths no longer than $|w|$, M' rejects w. **Verify**: for any $w \in L$, since M will enumerate w sometime, then M' will accept it. For $w \notin L$, when M has enumerated all strings of lengths no longer than $|w|$, M' will reject correctly. Thus M' indeed decides L. That is, L is recursive.

   " $\Rightarrow$ " Suppose L is recursive, that is, there exists a Turing machine M' that decides L. We can construct an enumerating Turing machine M as follows: using a non-decreasing order of lengths of strings and within the same length, using lexicographical order, we iterate each string, and for each string w, we run M' on w; when M' accepts it, M enumerates it; otherwise M skips it and continues. **Verify**: for any string $w \in L$, it will be accepted by M' and thus enumerated by M. And the order of enumeration is clearly in a non-decreasing order of lengths. For any string $w \notin L$, since it is not accepted by M', M will not enumerate it. Thus M indeed enumerates all strings of L in a non-decreaasing order of their lengths.

4. a) **Solvable:** Since within 10 steps, a deterministic TM can at most probe 10 squares. As a result, we can enumerate all the possibilities. In detail, we enumerate all the possible input string w, and for each input string, we simulate M on w at most 10 steps. If M accepts w, we stops the enumeration and output yes; else continues to enumerate. If M doesn't accept any input after the enumeration of all the possibilities, we output no.

   Obviously, this naive algorithm will halt within $10 \times |\Sigma|^{10}$ steps and correctly output whether there is a string w such that M accepts w within 10 steps.

   b) **Undecidable:** We've known that, given a TM M, it's undecidable whether $L(M) \neq \emptyset$ (we call it non-*Empty* problem). We will reduce from the non-*Empty* problem to this problem.

   Suppose, for any given deterministic TM M, there is a TM M_{10} (or an algorithm) that decides whether there exists a string w such that M accepts w in at least 10 steps.

   Construct a TM (or an algorithm) M_{non} that can decide the non-*Empty* problem:

   Input: any TM M

   Transform: construct another TM $M^{'}$ that behaves as follows: it first moves right 5 steps and move backward 5 steps, then simulates M on its corresponding input x. (We can represent $M^{'}$ as

   $$> R^5 L^5 \rightarrow M$$

)

   Note that if M accepts the input x, M' accepts x in at least 10 steps. If M rejects x, M' also rejects x.

   Invoke: run M_{10} on input $M^{'}$

   Output: if $M_{10}(M^{'})$ says yes, return yes; else return no.

   Verify: if $L(M) \neq \emptyset$, then $M_{10}(M^{'})$ says yes. Moreover, M_{non} accepts M.

   if $L(M) = \emptyset$, then $M_{10}(M^{'})$ says no. Moreover, M_{non} rejects M.

   As a result, M_{non} actually can decide non-*Empty* problem.

   But as we already know, non-*Empty* problem is undecidable. Thus a contradiction occurs. Then we can derive that the assumption is wrong. In other words, given a deterministic TM M, it's undecidable whether there exists a string w such that M accepts w in at least 10 steps.