Assignment 6

1. Prove that the following decision problem is undecidable by reducing it to a known unsolvable problem. Be sure to *clearly define your reduction.* It should be written like a pseudocode program that is straightfoward to implement. Two different TAs reading it should have the same understanding of how your reduction works.

   (a) Given a TM T and a word w, does T accept w in an even number of moves?

   solution: we reduce to the problem of does T accept w. Given a T, we preprocess it to T' where for every state q of T, we create q_o for "odd" parity and q_e for "even" parity. We make the start state q_0 be $q_{0,e}$ as the start state has done zero moves and is therefore an even number. We make $h_{a,e}$ the halt and accept state, and $h_{r,e}$ the halt and reject state. We set $\delta(h_{r,o}, \star) = (h_{r,e}, \star, \rightarrow)$ and $\delta(h_{a,o}, \star) = (h_{r,e}, \star, \rightarrow)$. This means that T' can only halt if it has executed an even number of moves. Effectively this creates an additional step at the end of processing the word, which is only done if the machine has thus far done an odd number of moves. Similarly, it only enters this halt and accept state if it T accepts w.

   ```
   bool T_accepts_w(TM T, WORD w) {
       create T' that keeps track of parity and only halts in even steps.
       return T_accepts_w_in_even(T', w);
   }
   ```

   l

   (b) Given a TM T, a word w, and a state q such that $q \neq h_a$ and $q \neq h_r$, does T ever enter q when processing w.

   solution: we reduce to the problem of does T accept w. Given a T, we preprocess it to T' where we insert a new state q. For any (p, a) such that $\delta(p, a) = (h_a, a', d)$ we make $\delta(p, a) = (q, a', d)$. We further add, for all $\sigma \in \Gamma$, $\delta(q, \sigma) = (h_a, \sigma, \rightarrow)$.

   Thus, the only path to h_a must transit through q, and no other processing will enter q. That is, the old machine enters h_a if and only if the new machine enters q.

   ```
   bool T_accepts_w(TM T, WORD w) {
       create T' that has new state q
       return T_enters_q_when_processing_w(T', w, q);
   }
   ```

   (c) Given a TM T, and two words w and x, does T accept either wx or xw?

   We reduce this to does T accept ϵ by passing ϵ for both w and x.

```
bool T_accepts_epsilon(TM T) {
    return T_accepts_either_wx_or_xw(T, epsilon, epsilon);
}
```