Max Shi
CS 334
November 14, 2019
I pledge my honor that I have abided by the Stevens Honor System

Problem Set 8

1.
    a. Let E be an enumerator for all possible strings. Thus, let us create a TM D that takes the input M as a Turing Machine and uses E. For every string that E prints out, run M on that string. If M accepts any string, accept. Thus, if M accepts some string, M will eventually accept a string that E prints out, as E will encompass all possible strings. Thus, this TM D recognizes NONEMPTY.
    b. We will prove this in a fashion very similar to the in-class proof that E_{TM} = {<M>: L(M) = ϕ} is undecidable. Let TM N be a TM that decides NONEMPTY. We will create a TM that decides A_{TM}, to prove that N cannot exist. Let this TM be A. On an input of a TM M and a string w, let A construct a TM M_1 that takes an input x, and if x is w, it runs M on x and accepts if M accepts. In any other case (x does not equal w, M does not accept x), M_1 rejects. Then, run N(M_1) and let the output of N also be the output of A. Therefore, if M accepts w, the language of M_1 is nonempty, as w is in the language of M_1, and N accepts. If M does not accept w, then the language of M_1 is empty, and N rejects. Thus, A decides A_{TM}, but because A_{TM} is undecidable, N cannot exist, and NONEMPTY is not decidable.
2.
    a. Let E be an enumerator for all possible strings. Then, let us create a TM D that takes a TM M as input and uses E. Then, let D keep a count of the number of accepted strings as it begins to run M on each string that E prints out. If the count gets to 17, let D accept. (This can be done by reserving the first 17 spots on the tape and marking them off until they are all full.) Thus, if M accepts at least 17 strings, D will accept, and if it does not, then D will never accept. Thus, D recognizes L_{17}.
    b. Let TM L be a TM that decides L_{17}. Let us construct a TM A that takes in a TM M and a string w as input. Let A construct a TM M_1 that takes as input a string x. On input x, M_1 will run M on input w, and accept if M accepts w and reject otherwise. Thus, if M accepts w, then the language of M_1 will be all strings, and if it does not, then the language of M_1 will be empty. Then, run L(M_1). If L accepts, then let A accept, and if it rejects, let A reject. Because the language of M_1 is all strings if M accepts w, it is in the language of L, as infinity>17, and if the w is not accepted by M, the language of M_1 will be empty, and 0<17, so L will reject. Thus, A decides A_{YM}, and because A_{TM} is undecidable, then L cannot exist. Therefore, L_{17} is undecidable.
3. The result of running Ben's program is no result, as the machine never accepts nor rejects. However, this is still a TM, it just so happens that this TM has an empty language. This exposes the fault in the reasoning: this TM would contradict itself and not be able to exist if it ever got to the point where it decides whether to accept or reject. However, this TM will always be stuck on step "2a," which is running B(x).

4. If D accepts <X> then simulate <N> on w.
   If D rejects <X> then simulate <M> on w.
   We conclude that D cannot exist, and the language is undecidable.