1. Certain assemblers form their integer literals in the following way. **Binary** literals consist of one or more binary digits (0, 1) followed by the letter B; e.g., 10110B. **Octal** literals consist of one or more octal digits (0 through 7) followed by the letter Q (since O looks too much like 0); e.g., 1234567Q. **Hexadecimal** literals consist of at least one decimal digit (0 through 9), followed by zero or more hexadecimal digits (0 through 9 and A through F) followed by the letter H; e.g., 0ABCDEFH. **Decimal** literals consist of at least one decimal digit optionally followed by the letter D; e.g., 1234.

   (a) Draw the state diagram of an NFA (not a DFA) for these literal forms; you may use $\epsilon$-transitions, you don’t need to use Thompson’s construction.

   (b) Give a regular expression for the literals; you may use $\epsilon$.

   (c) Modula-3 based literals have the form

   \[
   \text{base}\_\text{digits}
   \]

   (e.g., 2_11111111, 8_377, 16_FF, 255 are equivalent integer literals). Why is the Modula-3 form preferable to the assembler form introduced here, from the standpoint of a compiler (or assembler) writer?

2. (a) Using Thompson’s construction, construct an NFA that recognizes the same language as defined by the following regular expression:

   \[
   (1*01*0)*1*
   \]

   (b) Using the subset construction, convert the NFA into a DFA. Optimize the resulting DFA by merging any equivalent states.