1. (6 pts) OCaml Types and Type Inference

   a. (2 pts) Give the type of the following OCaml expression
      \[ 4 :: [ ] \]
      \text{Type} = \text{int list}

   b. (2 pts) Write an OCaml expression with the following type
      \[ \text{int} \times \text{string} \]
      \text{Code} = (1,"foo")

   c. (2 pts) Give the value of the following OCaml expression. If an error exists, describe
      the error.
      \[ \text{let } x = 3 \text{ in let } x = 5 \text{ in } x \times x \]
      \text{Value} = 25

2. (4 pts) Regular expressions and NFA

   Given the following NFAs representing the regular expressions \( R \) and \( S \), use the algorithm
   presented in class to add/modify states and transitions to produce a NFA representing the regular
   expression \( R \times S \).

   ![Diagram of two NFAs with transitions labeled by \( \varepsilon \) and input symbols]

   3. (10 pts) NFA and DFA

   Consider the following NFA

   a. (2 pts) Does the NFA accept the string “ab”? If it accepts the string, list a sequence of
      state transitions (e.g., 3,2,1) that leads to acceptance of “ab”.
      \text{Yes. 1,2,3,1,4,3,1,2}

   b. (8 pts) Convert the NFA to a DFA using the subset construction algorithm discussed
      in class. Be sure to label each state in the DFA with the corresponding state(s) in the
      NFA.

   ![Diagram of the NFA and the corresponding DFA]