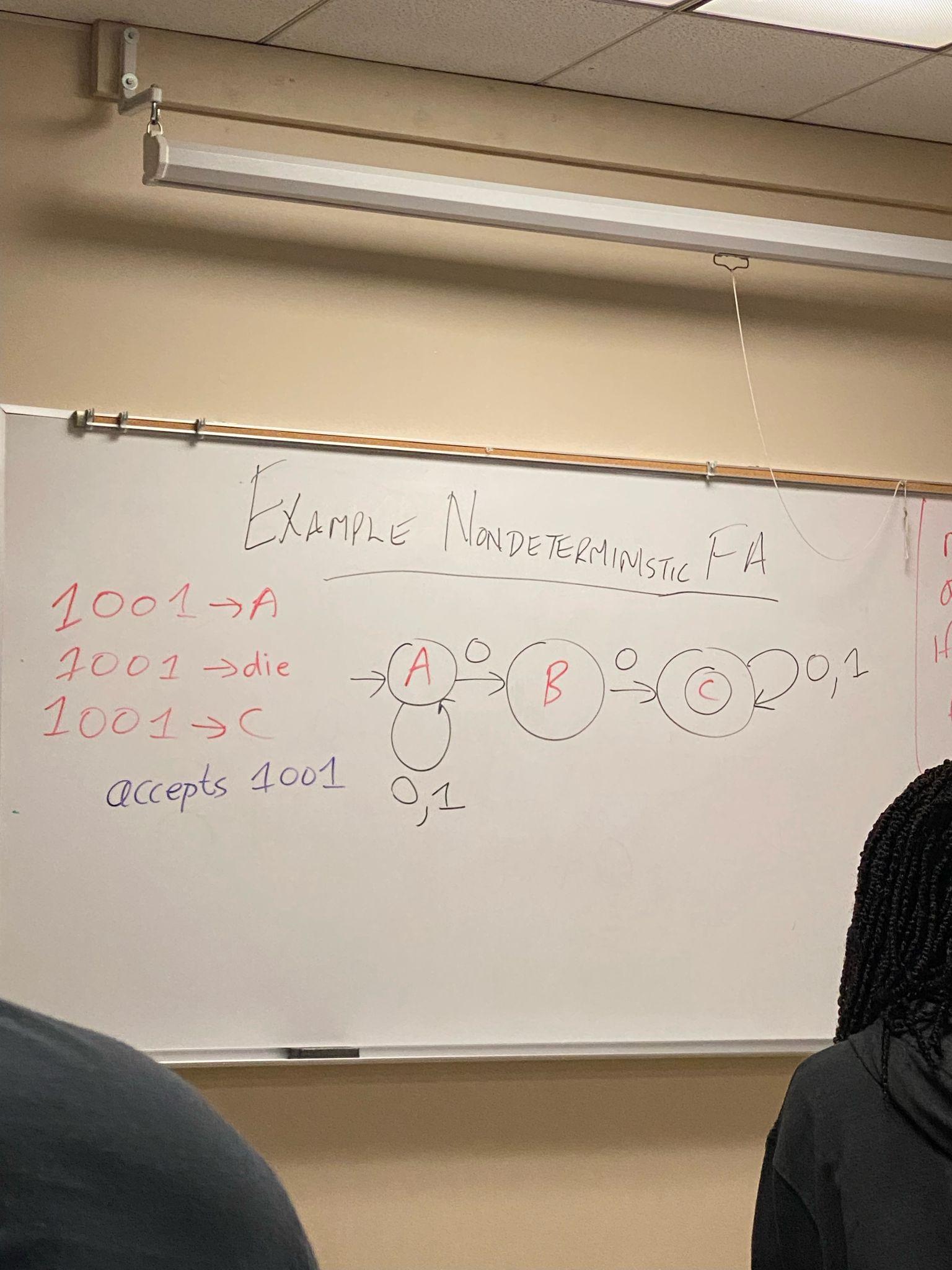
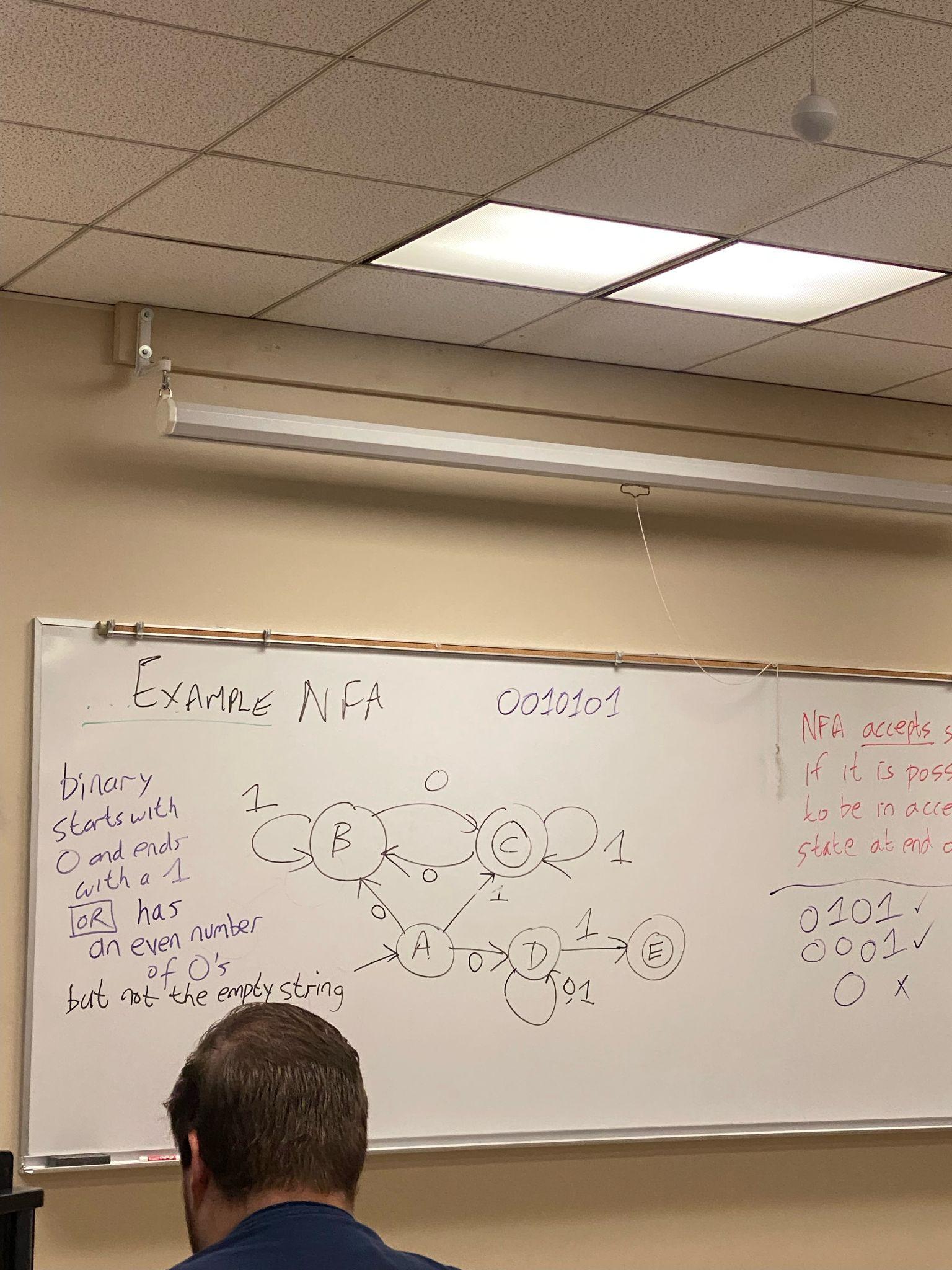
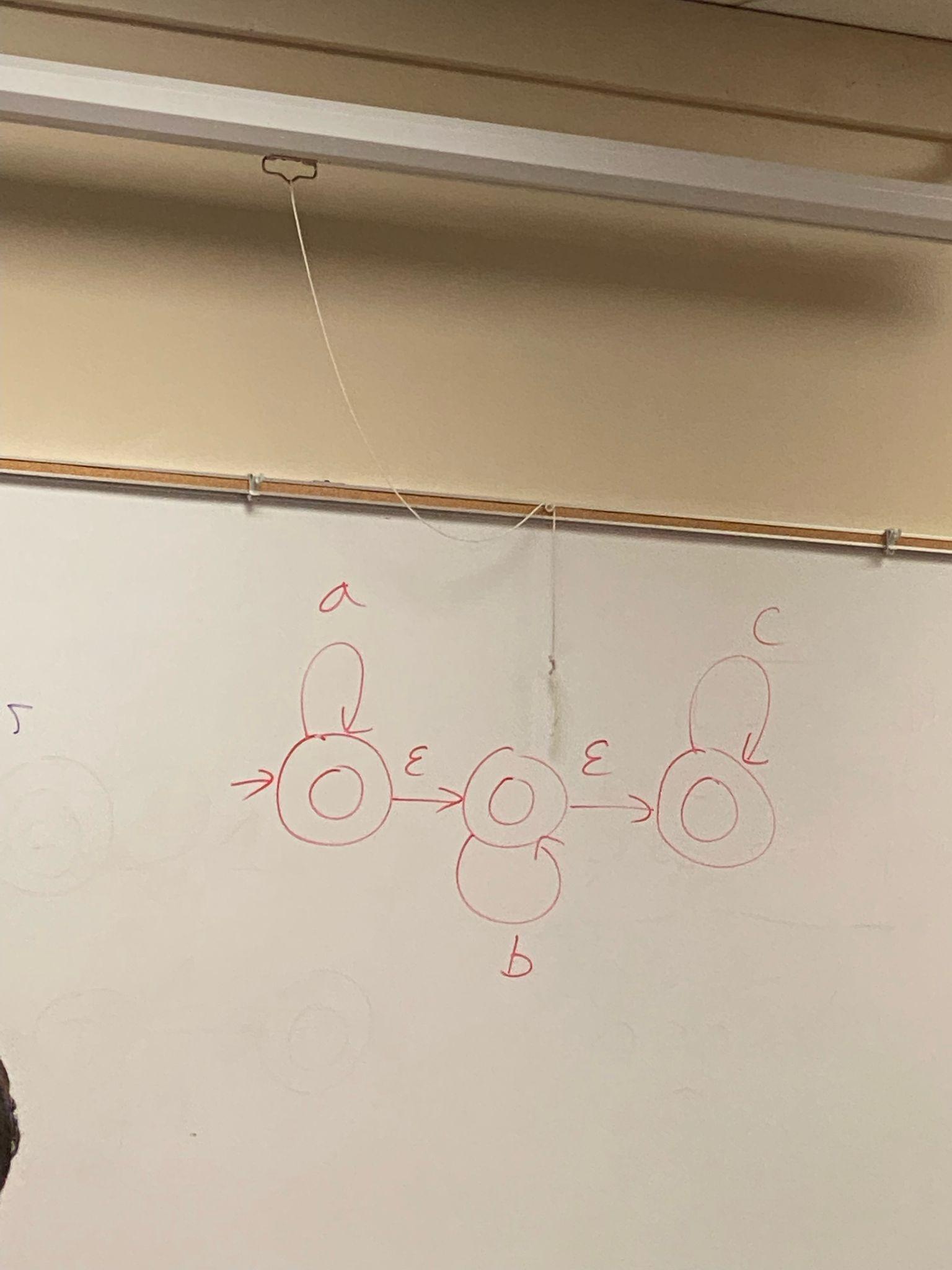
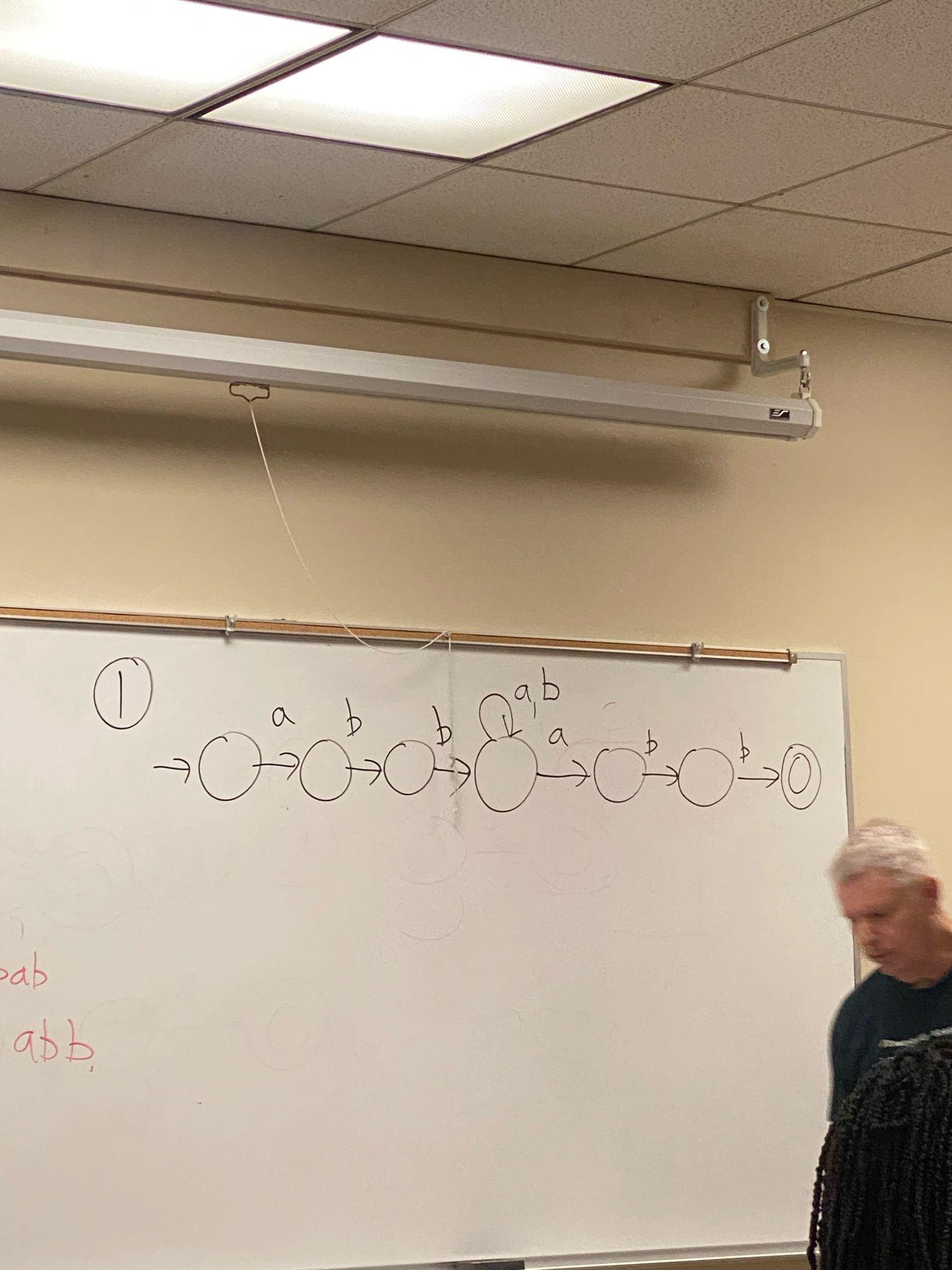
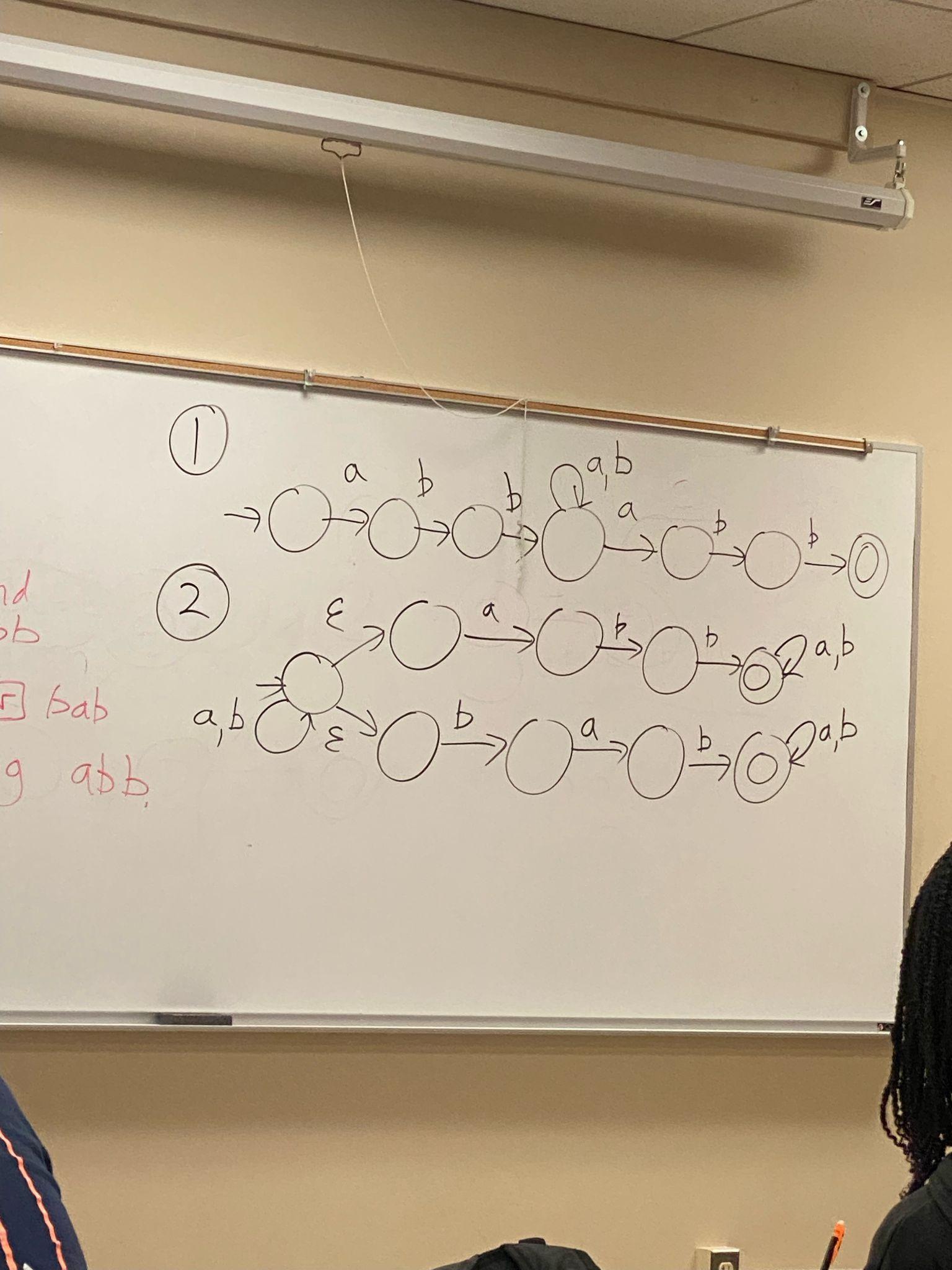
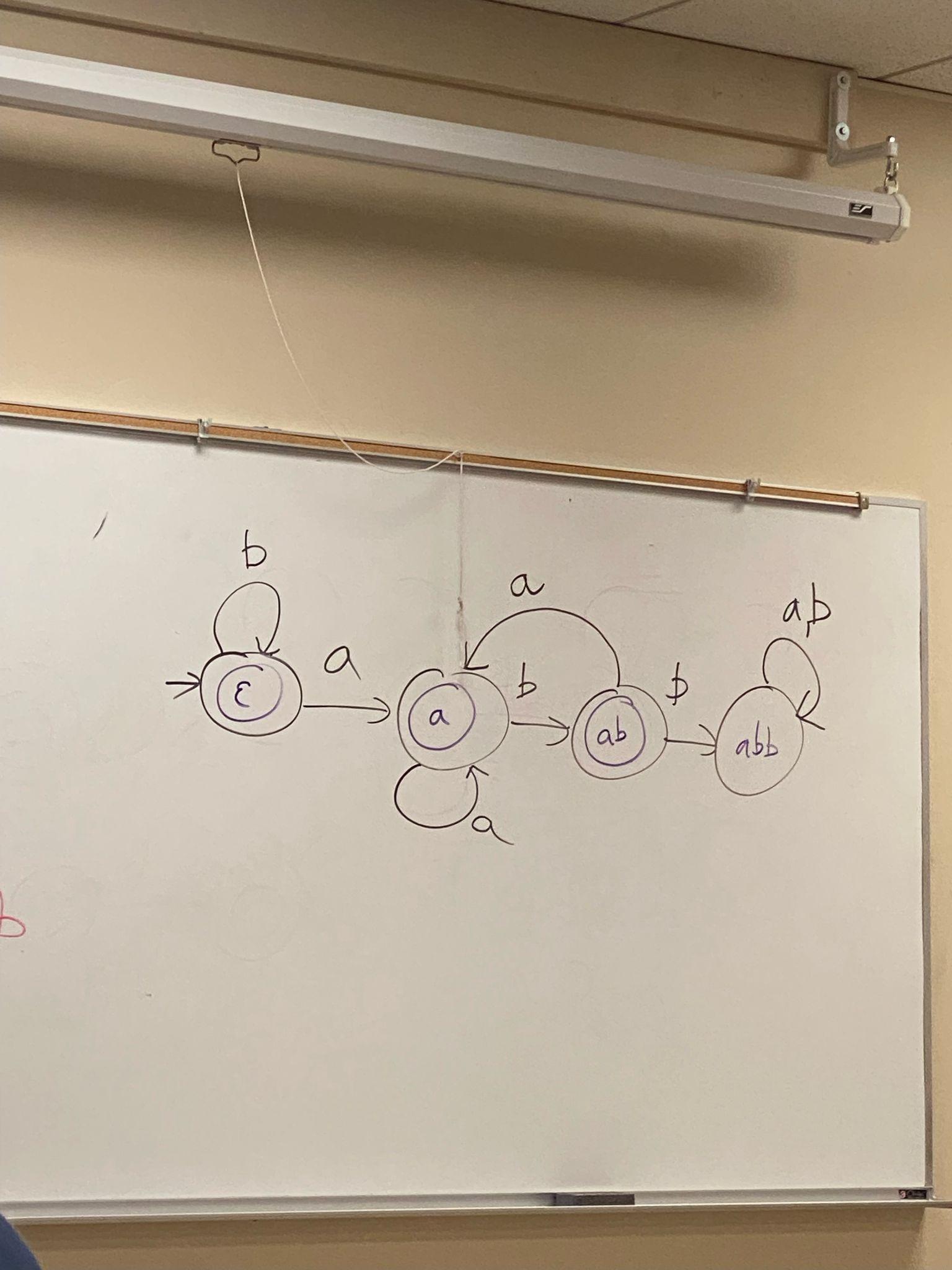
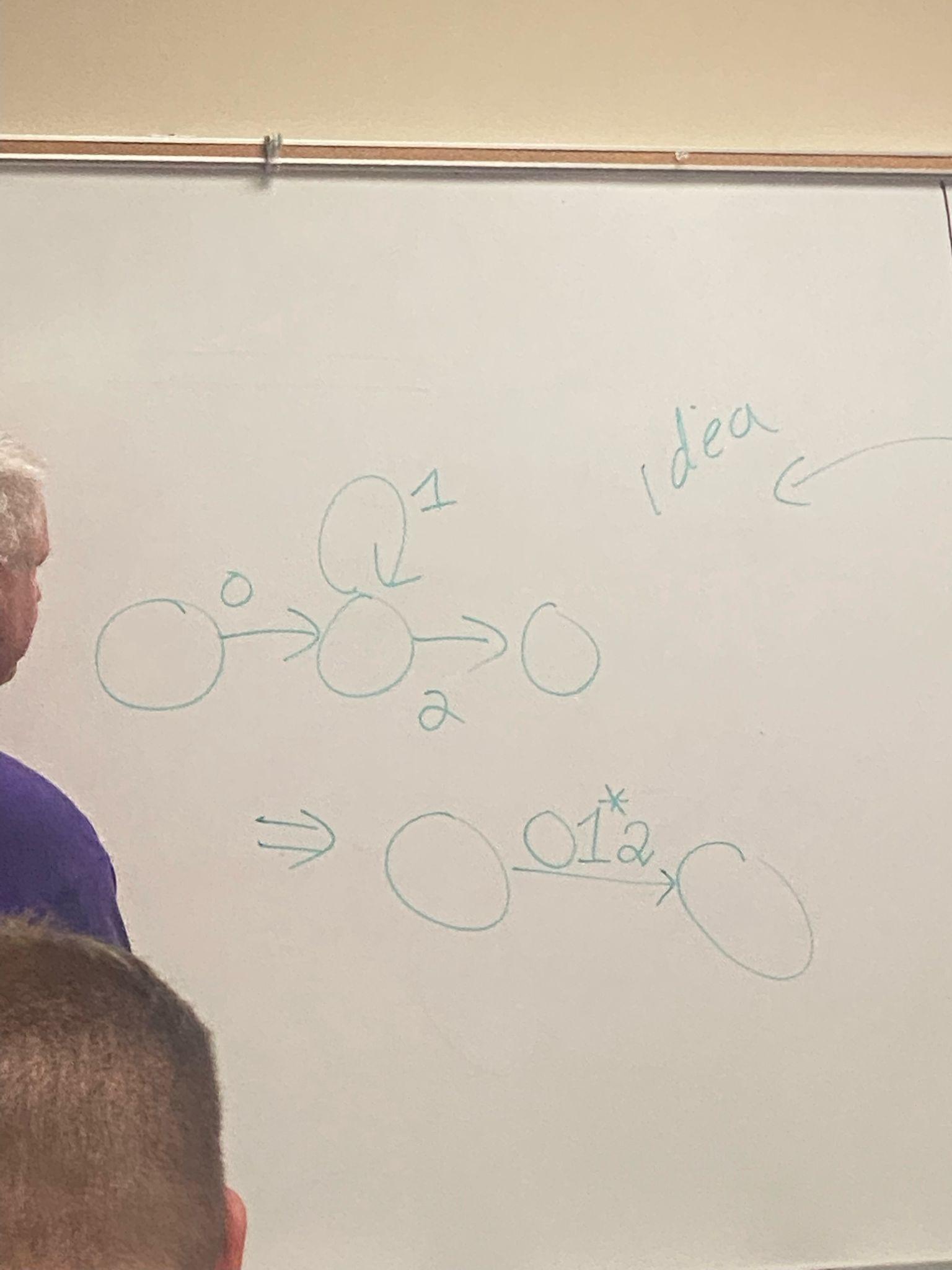
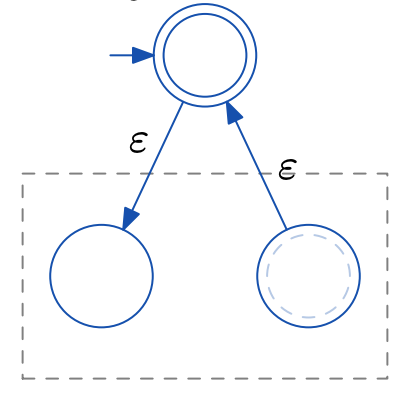
Nondeterministic Finite Automata

* Nondeterministic Finite Automaton (NFA)
  + Accepts a string if it is possible to end in an accept state.
* Example of a Nondeterministic Finite Automaton (NFA):
  + - Nondeterminism will allow us to have two transitions of the same name.
    - Nondeterminism will allow us to have only one transition for one state.
    - There is not a symmetric definition.
* Epsilon means an empty string
* Assignment 2
  + Build a DFA that accepts all decimal numbers
  + Dying does mean reject!
* NFA accepts the string if it is possible to be in an accept state at the end of the string.
* Shows up in two ways
  + Idea that there are multiple possibilities (ex. 1 can be used twice out of a state)
  + We can avoid drawing reject traps, not required to have a transition.
* NFA’s can be seen as **clairvoyant**
  + It knows what it needs to do in order to have a string accepted.
* Epsilon Transitions
  + An epsilon transition allows an NFA to change state without consuming an input character.
  + The example to the right is an answer to having all strings be in alphabetical order.
  + They can be used even after the last character.
* In class practice 4 : NFA’s
  + 1) All strings beginning and ending with abb
  + All strings containing abb or bab (or both) as a substring



* + All strings NOT containing abb as a substring



* Give both a regular expression and a finite automaton for the following language. The alphabet is {a, b, c}. The language is all strings that satisfy at least one of the following properties:
  + (i) the string starts and end with an a,
  + (ii) the string starts with a b and has even length,
  + or (iii) there is exactly one c in the string.
* **EPSILON CAN ONLY BE IN NFA**
* **SIGMA CAN BE IN EVERYTHING**
* **EPSILON MEANS AN EMPTY STRING TRANSITION**
* **SIGMA MEANS EVERYTHING IN THE ALPHABET**
* Kleene’s Theorem
  + The following are equivalent
    - 1. There is a DFA for language L.
    - 2. There is a NFA for language L.
    - 3. There is an RE for language L.
  + If one of these statements is true, then all of them are true.
  + Three Algorithm
    - Convert FA to RE.
      * This can be represented by having an RE on the transition
    - Convert RE to NFA.
    - Convert NFA to DFA.
* RE to NFA
  + Recursion!!!!!!!
  + Need to be able to union, concatenate and star (\*) FA.
  + If you have an FA for RE’s R and S, you can build an FA for RE R+S.
    - You add a new start state and join it to the old start state of the two machines using epsilon transitions.
    - If you have an FA for RE’s R and S, you can build an FA for RE RS.
      * You can join them with an Epsilon transition.
  + The base case of the FA will be a single character.
  + If you have an FA for some RE R, you can build an FA for RE R\*
    - R\* means to allow the machine to cycle from the accept state back to the start state
    - Don’t make the start state an accept state because it can go through the start state multiple times.
    - To avoid this, build a new start state, which is the only accept state; then put ε-transition from it to the old start state, and from old accept states to it; and change every old accept state to reject.
* If there is at least one path that accepts the string, it is accepted.