**Compilers**

Lecture 2

Lecturer Dr. Zainab Hashim

Third stage – First semester

Computers Department

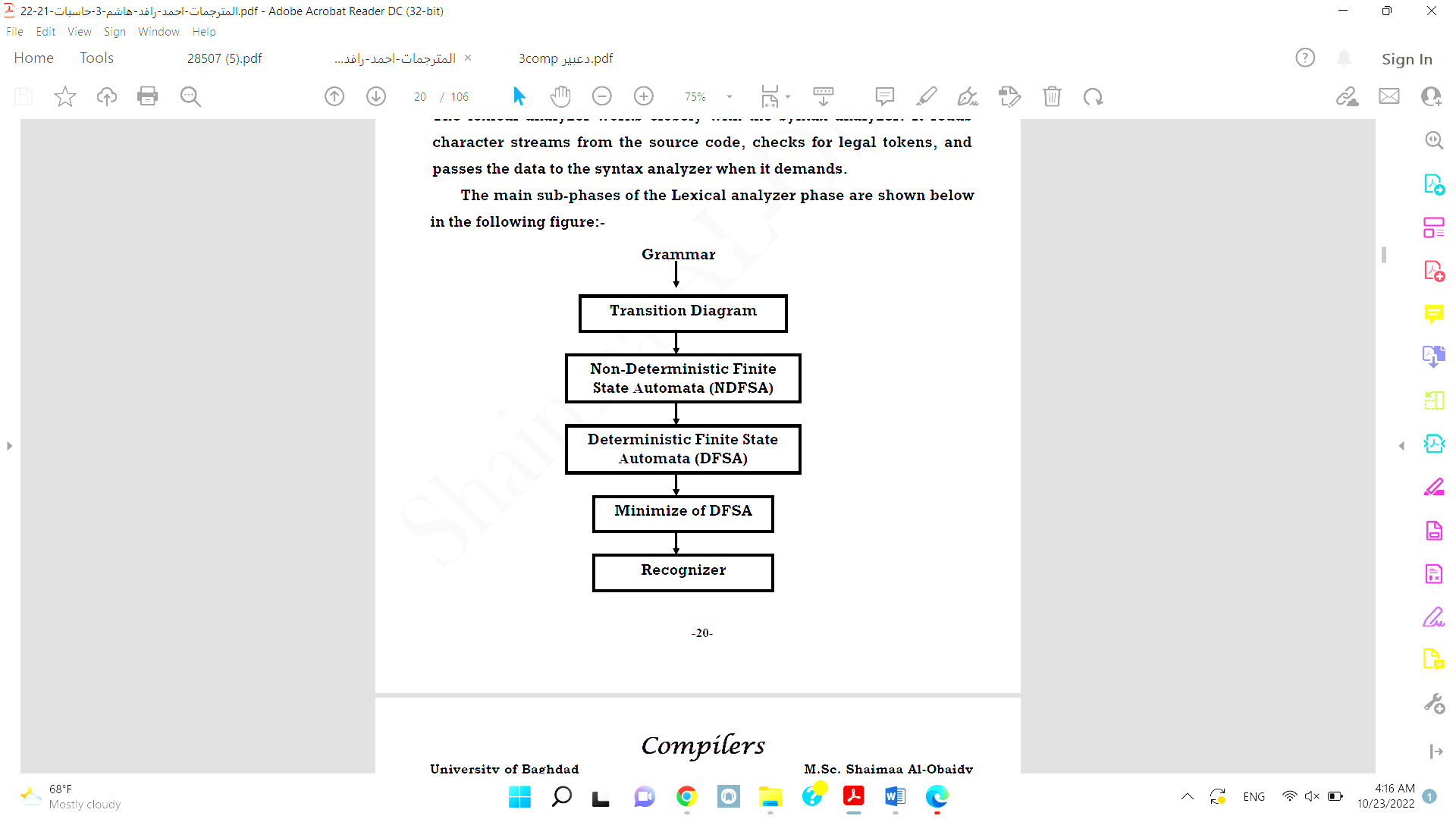
**2022-2023**

**Lecture2: *Lexical Analyzer Design***

Lexical analysis is the first phase of a compiler. It takes modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.

**The main sub-phases of the Lexical analyzer phase are shown below in the following figure:-**



1. The grammar will converted to a Transition Diagram using special algorithm.
2. The converted Transition Diagram must be checked whether if it is in NDFSA form or not; if so, the grammar must converted to DFSA using algorithm which will be described in this chapter.
3. The resulted grammar will be in DFSA form which must be minimized to reduce the number of nodes depending on algorithm designed for this purpose (fast searching and minimum memory storage).
4. The final sub-phase in lexical analyzer phase is to recognize if the input string or statement is accepted or not depending on a specific grammar.

***Finite State Automata* (FSA):-**

Is a mathematical model consists of:-

1. A set of terminal symbols

2. Transition functions

3. One-Initial state (Start state)

4. One or Set of Final states

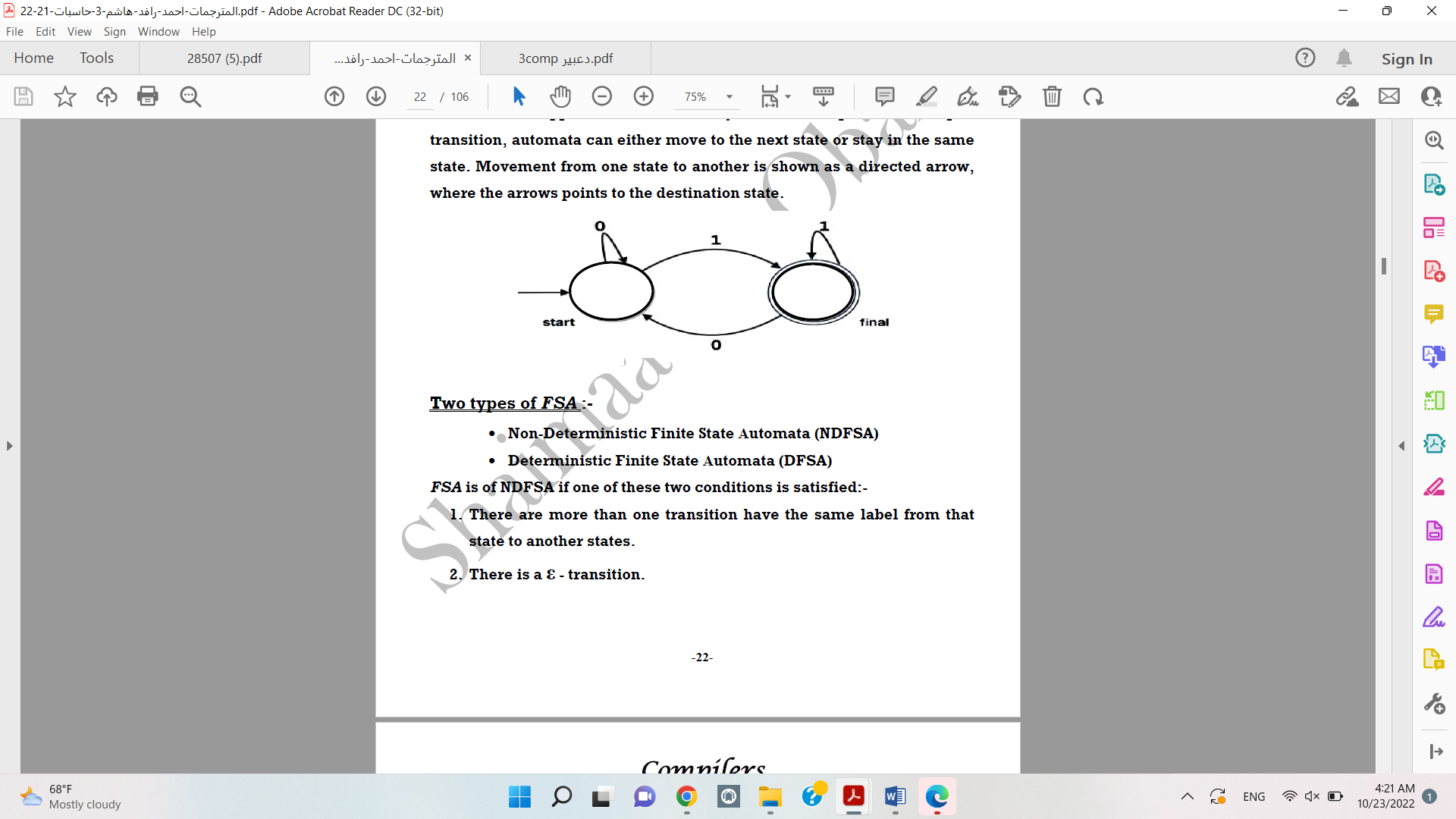
5. Finite set of elements called states

**States :** States of FSA are represented by circles. State names or numbers are written inside circles.

**Start state :** The state from where the FSA starts, is known as the start state. Start state has an arrow pointed towards it.

**Final State :-** If the input string is successfully parsed, the automata is expected to be in this state. Final state is represented by double circles, it is also called the Accepting State.

**A transition :-** Is denoted by an arrow connecting two states, the arrow is labeled by the symbol (possibly e). The transition from one state to another state happens when a desired symbol in the input is found. Upon transition, automata can either move to the next state or stay in the same state. Movement from one state to another is shown as a directed arrow, where the arrows points to the destination state.



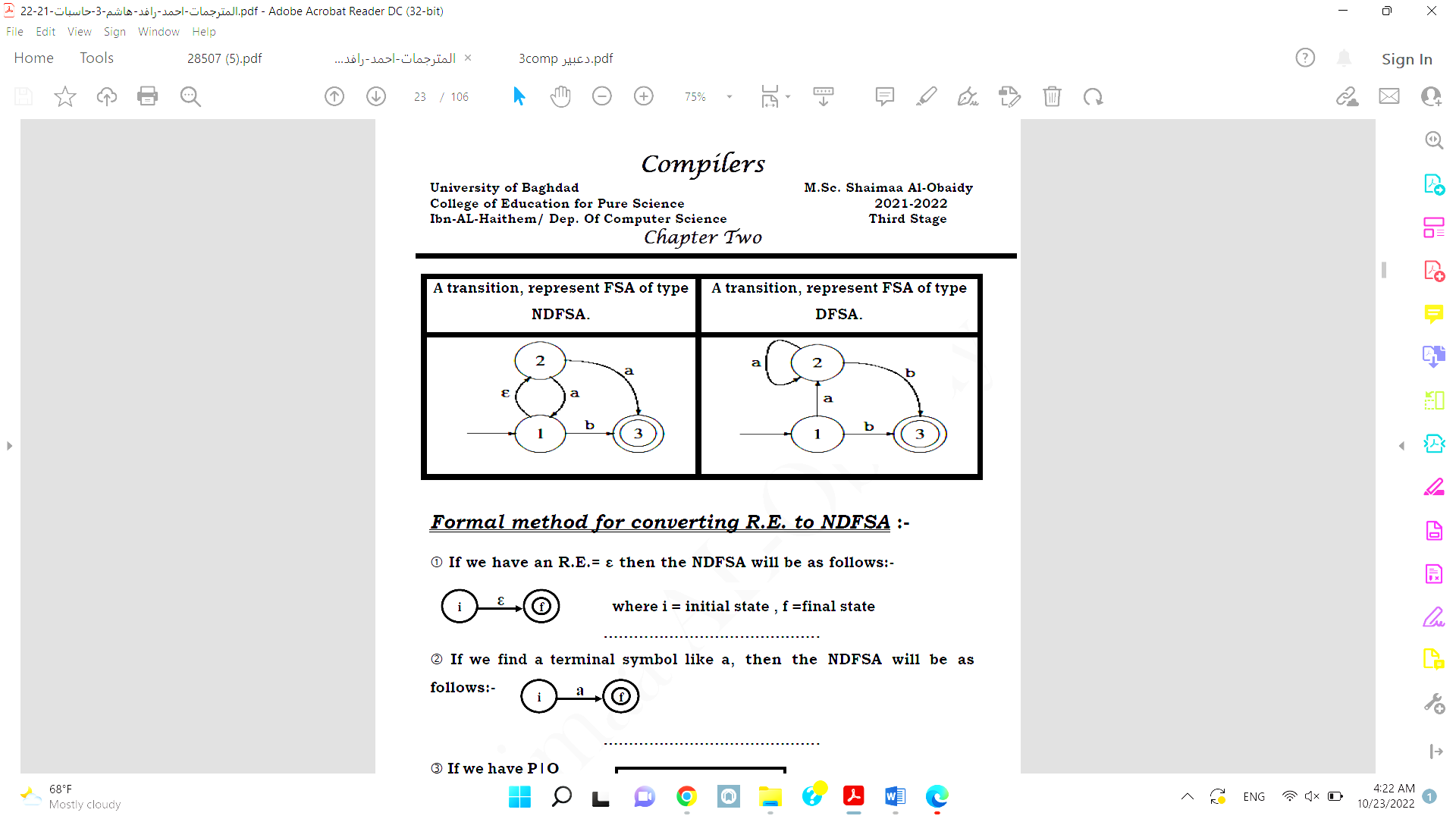
**Two types of *FSA*:**

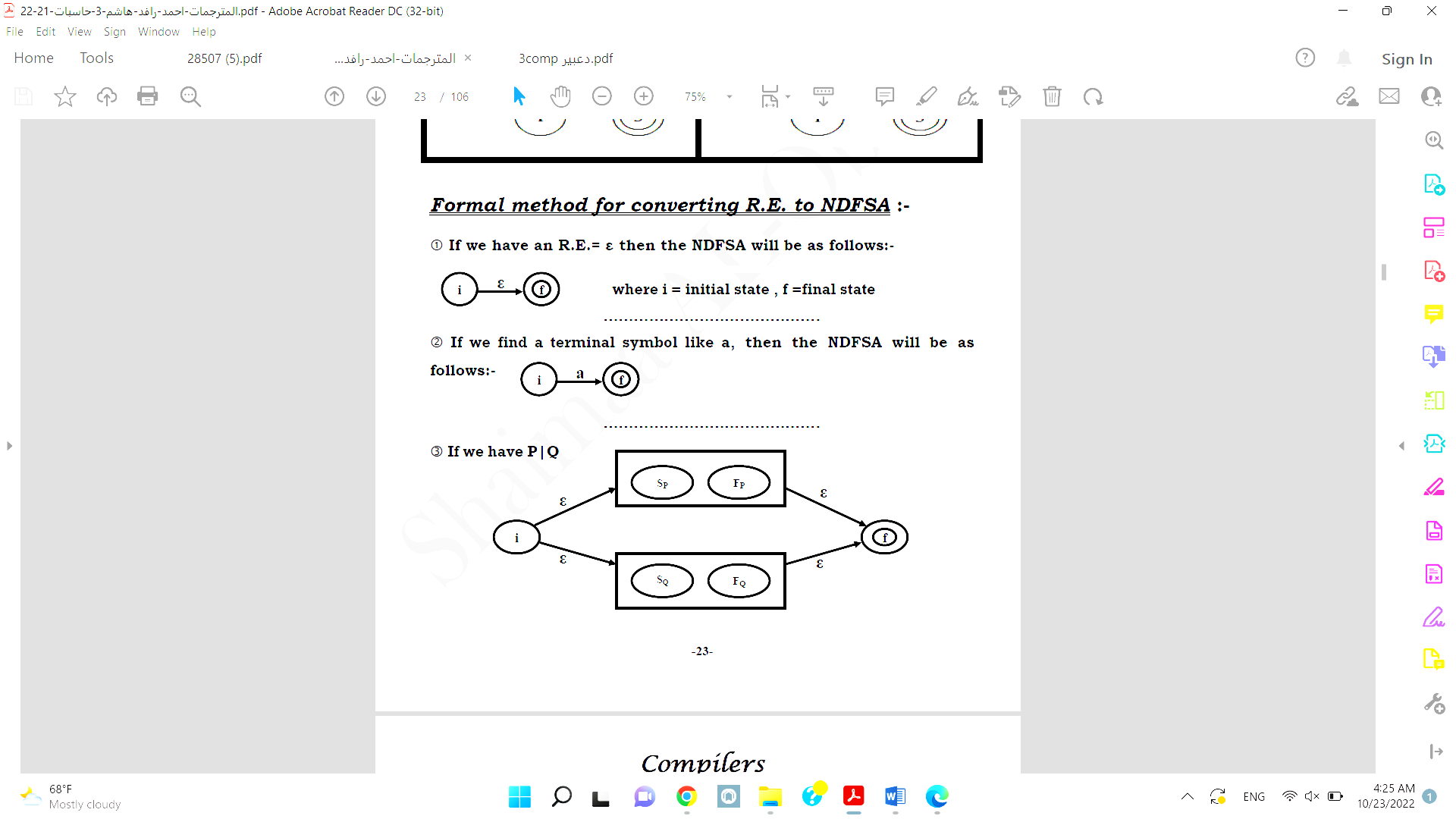
**Non-Deterministic Finite State Automata (NDFSA)**

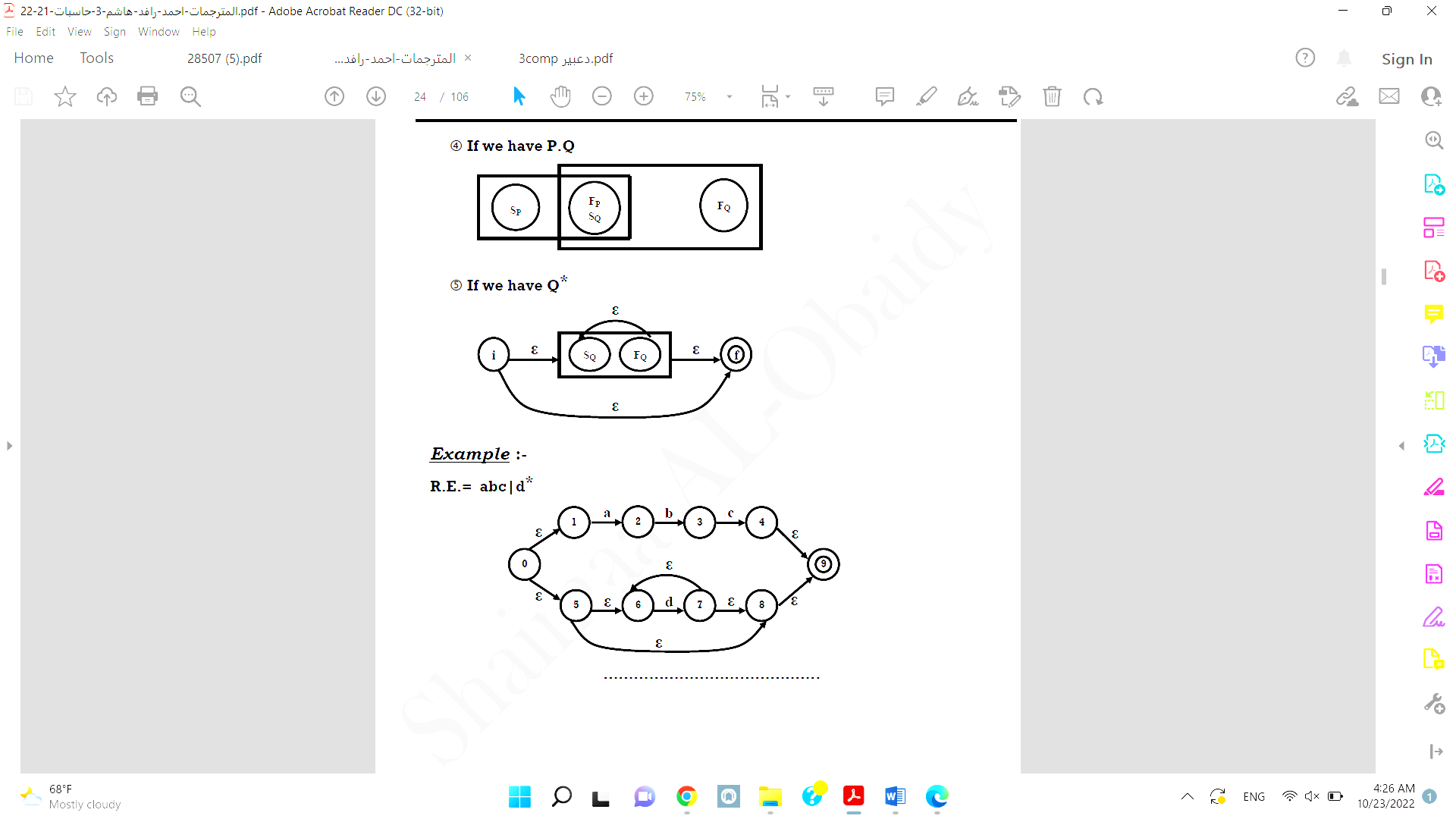
**Deterministic Finite State Automata (DFSA)**

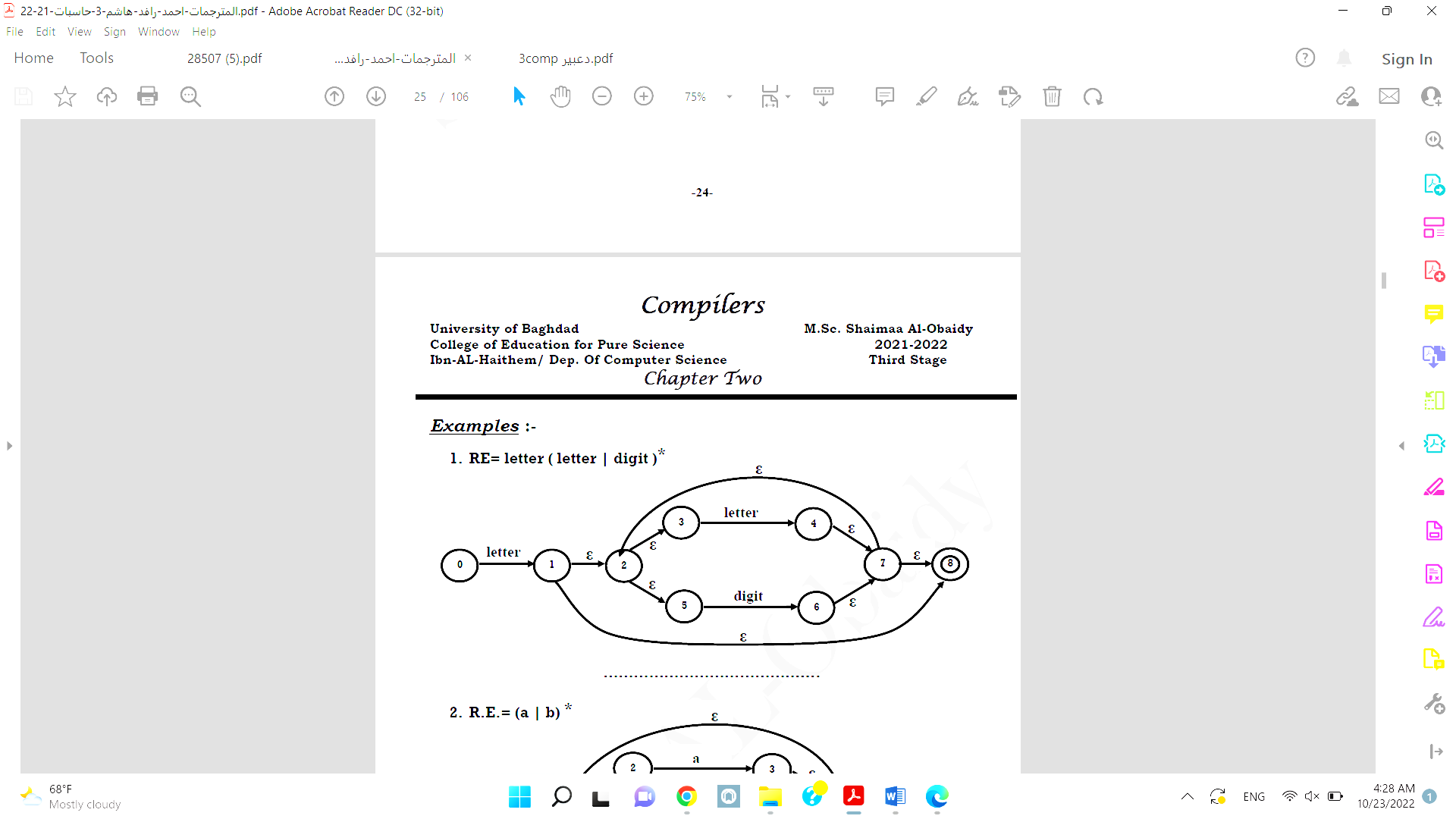
*FSA* is of NDFSA if one of these two conditions is satisfied:-

1. There are more than one transition have the same label from that state to another states.
2. There is a ε - transition.









**H.W) R.E.= (a | b) \***

***Algorithm for transforming NDFSA to DFSA*:**

**Initially let x=** ℇ***-*Closure ({S0}) marked as the start state of DFSA, S0 is the start state of NDFSA;**

**While there is unmarked states X = {S1, S2, ... ,Sn} of DFSA do**

**Begin**

**For each terminal symbol (a** ∈ Σ) **do**

**Begin**

**Let M be the set of states to which there is transition on *a* from some states Si in X ;**

**Y =** ℇ***-*Closure ({ M });**

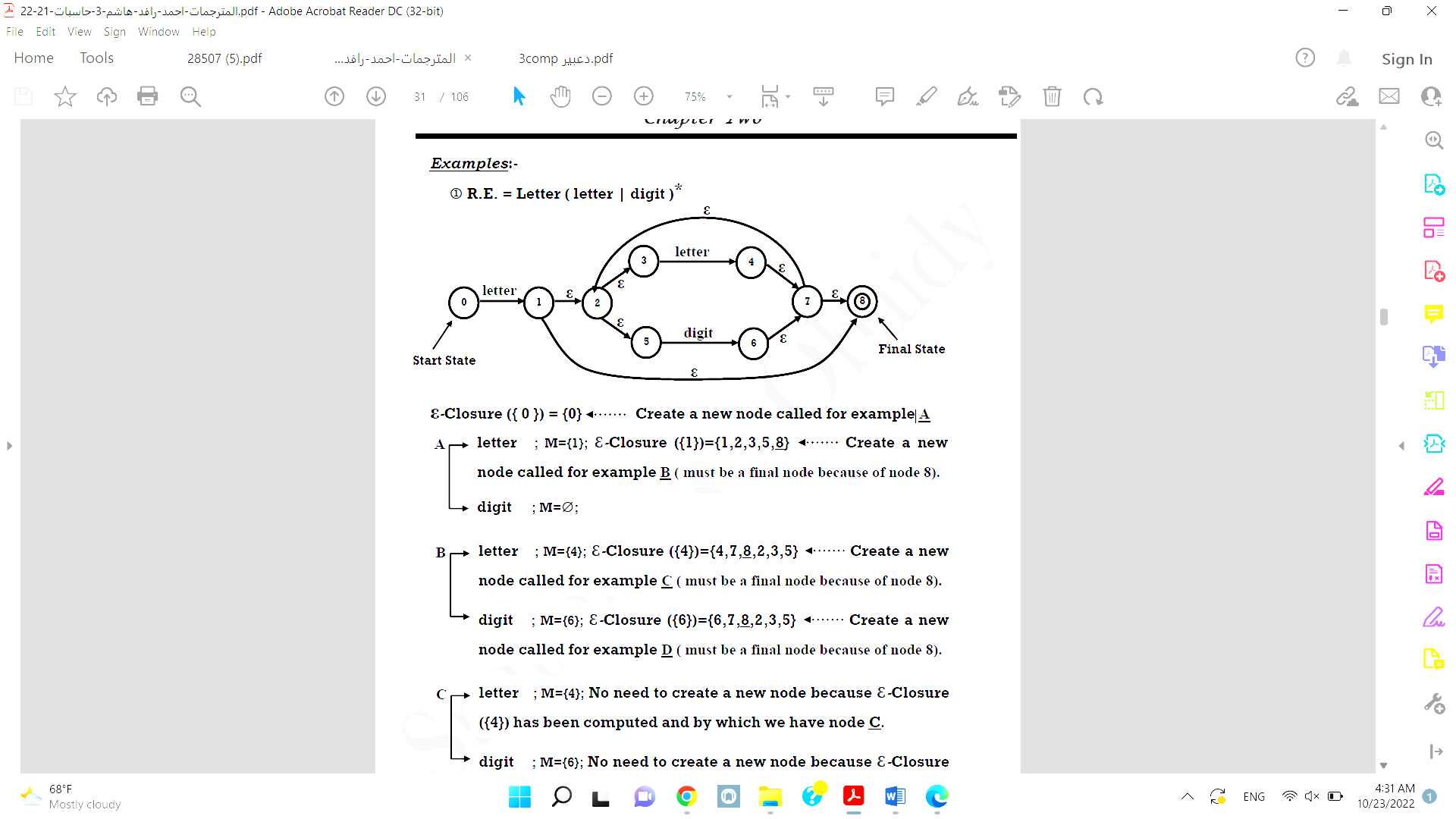
**If Y has not yet been added to the set of states of DFSA then make Y an unmarked state of DFSA;**

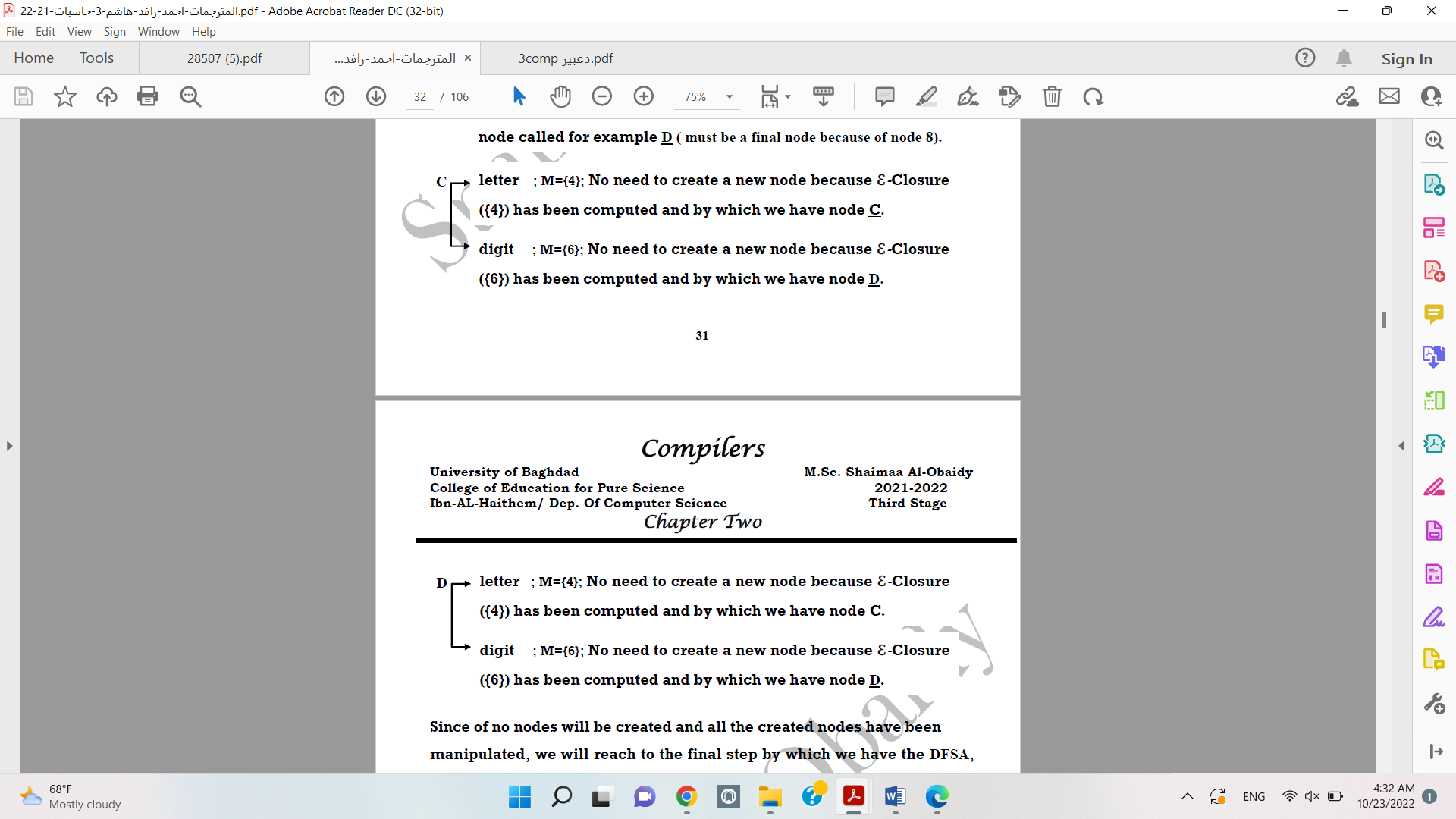
**Create an edge by adding a transition from X to Y labeled *a* if not present;**

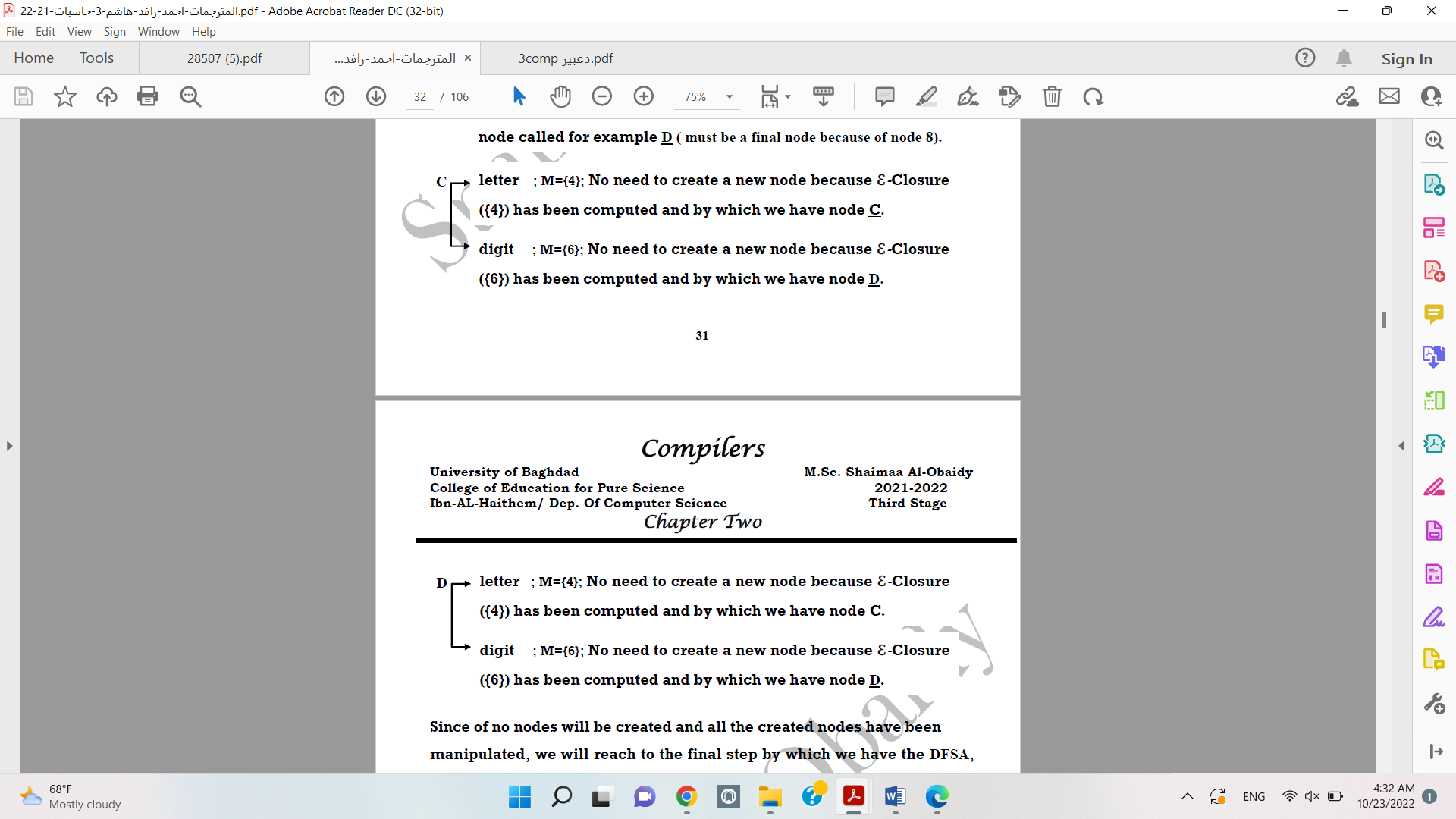
**End;**

**End;**

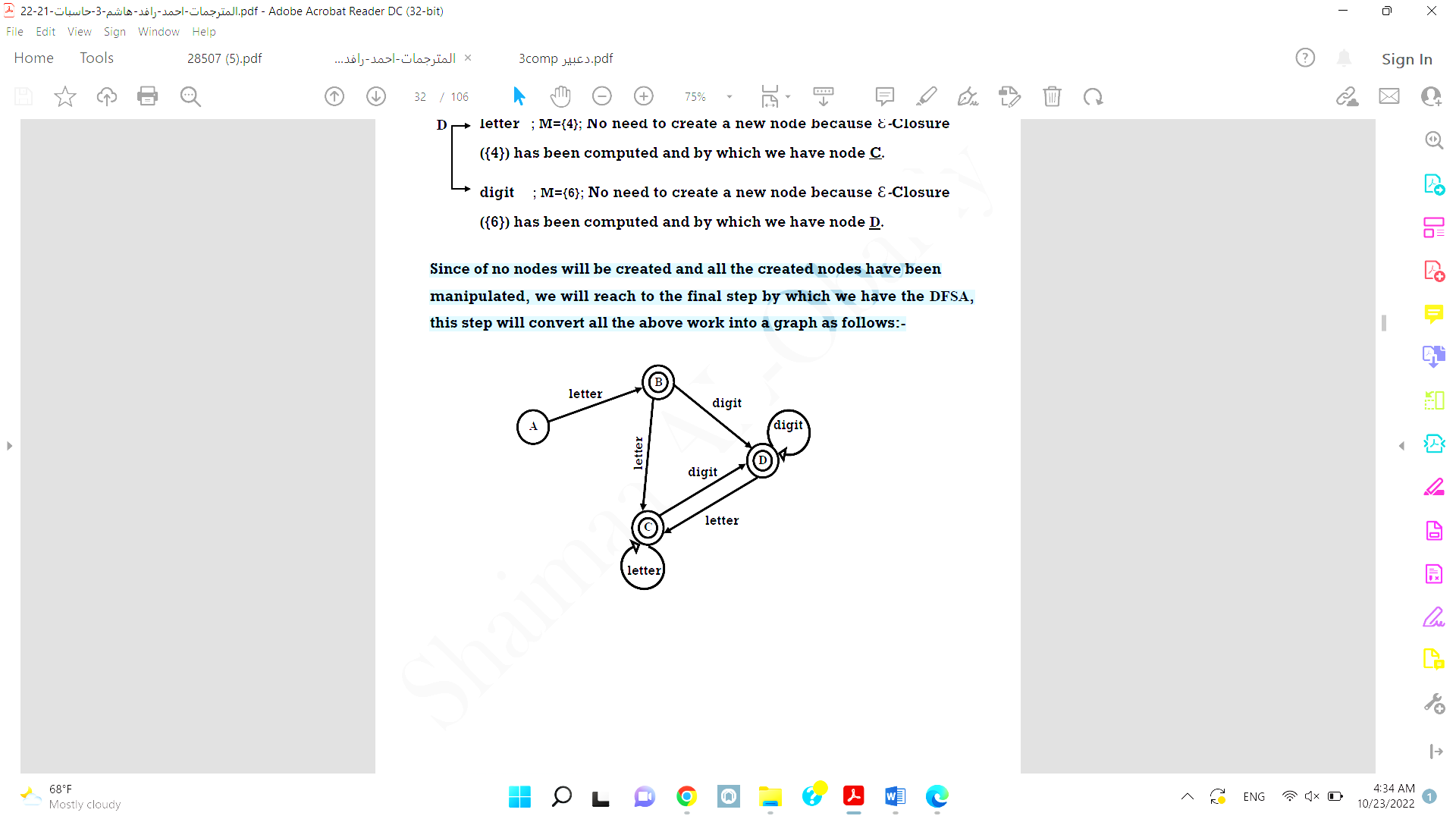
**End {algorithm}**







Since of no nodes will be created and all the created nodes have been manipulated, we will reach to the final step by which we have the DFSA, this step will convert all the above work into a graph as follows:-



***Minimizing of DFSA*:**

The purposes of minimization are:-

1. Efficiency.

2. Optimal DFSA.

***Algorithm*:-**

1. Construct an initial partition Л of the set of states with two groups: the accepting states F and the non accepting states S-F; where S is the set of all states of DFSA.

2. for each group G of Л do

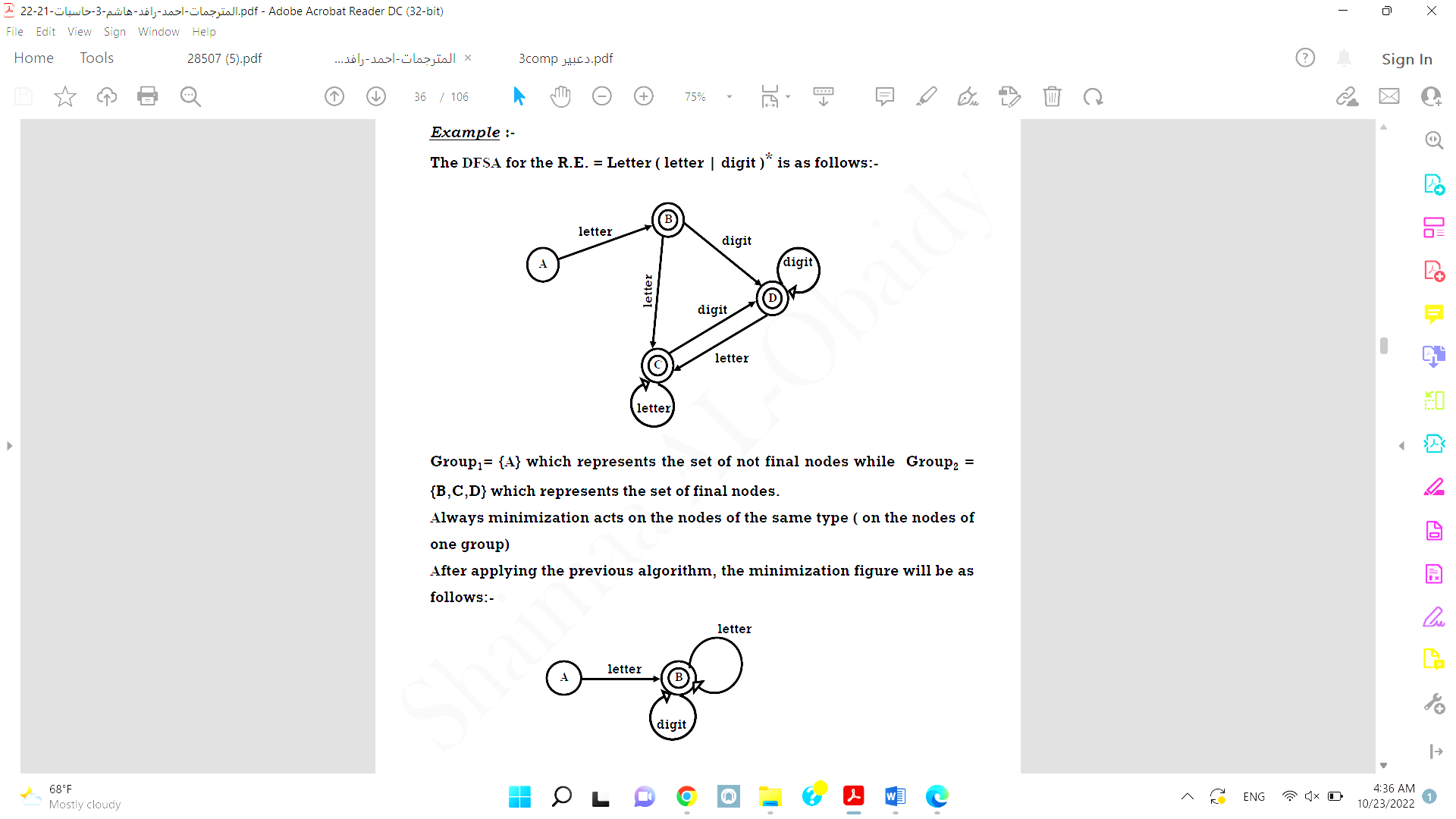
Begin

partition G into subgroups such that two states S and T of G are in the same subgroup if and only if for all input symbols *a*, and states S and T have transitions on *a* to states in the same group of Л, replace G in Лnew by the set of all subgroups formed .

End

3. If Лnew = Л, let Лfinal = Л and continue with step (4), otherwise repeat step (2) with Л := Лnew

4. Choose one state in each group of the partition Лfinal as the representative for that group.



**H.W)) Minimize the following DFSA:**

