1. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with a and end with a
2. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with 0 and end with 1
3. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0A1 A → 0A | 1A | ε

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S0 | 1S1 | 0 | 1 | ε

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

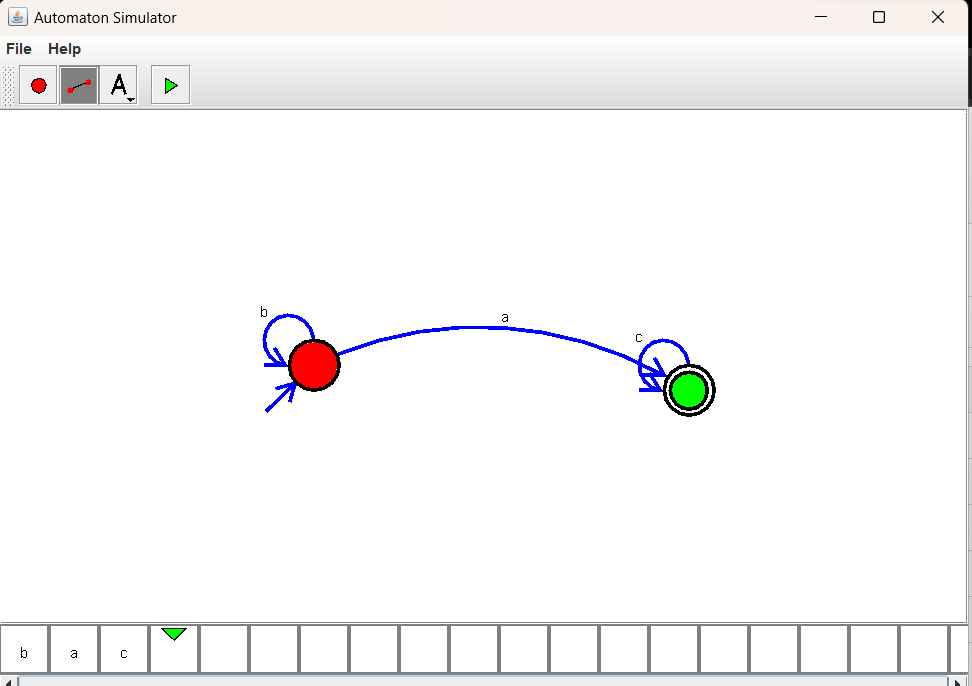
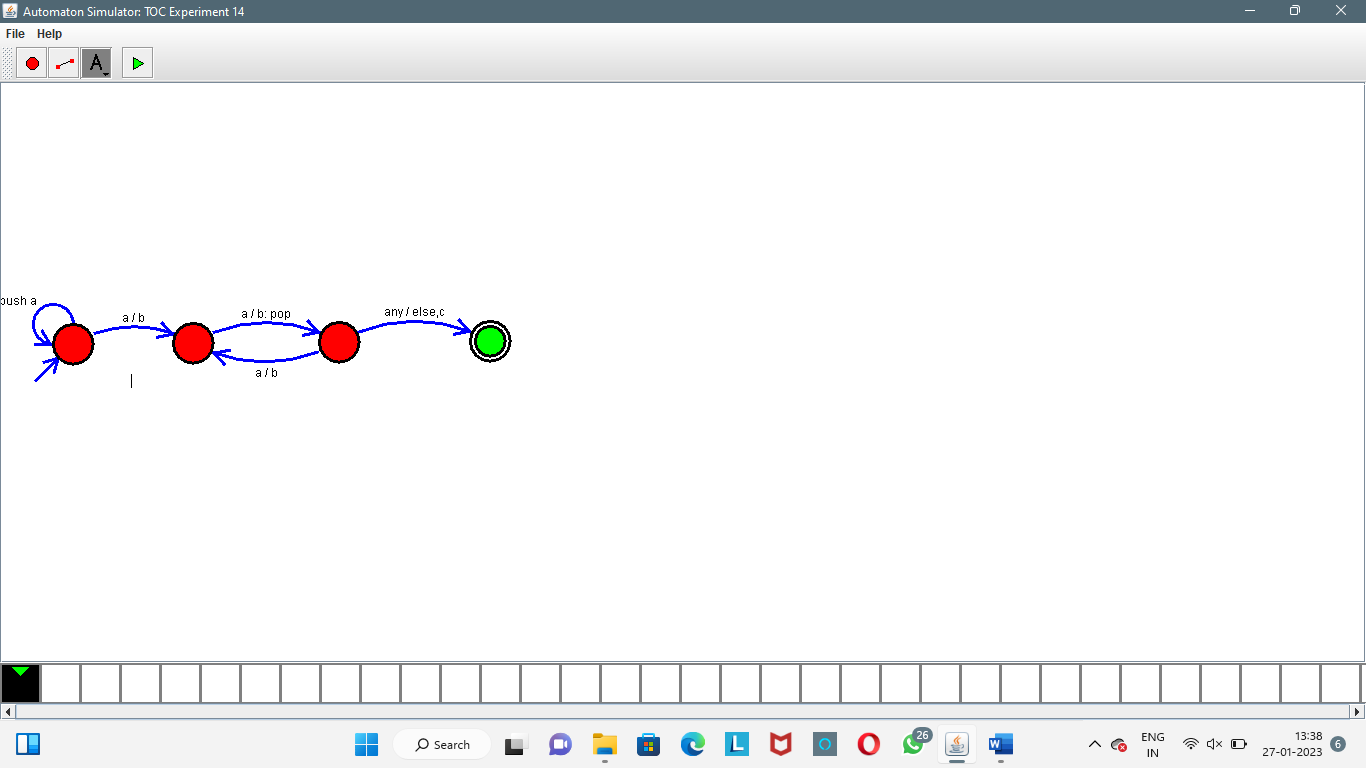
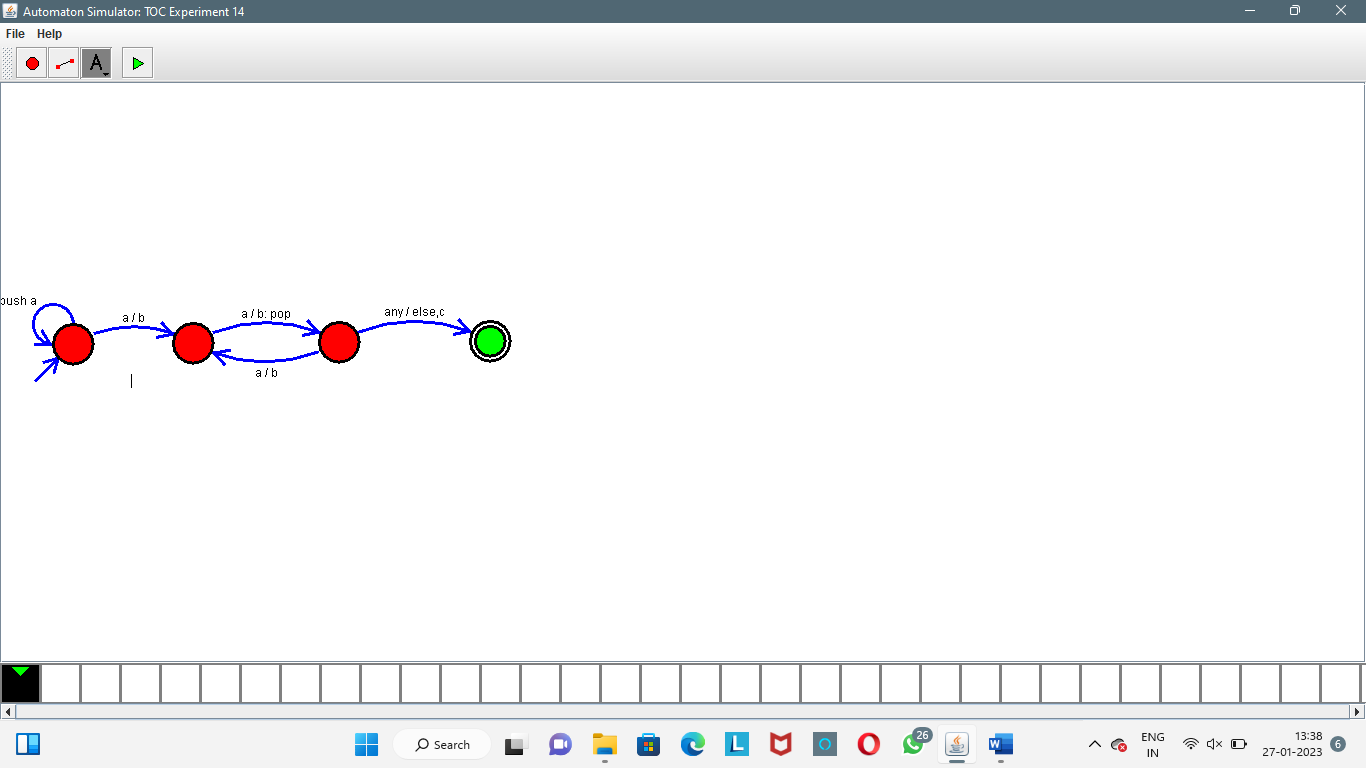
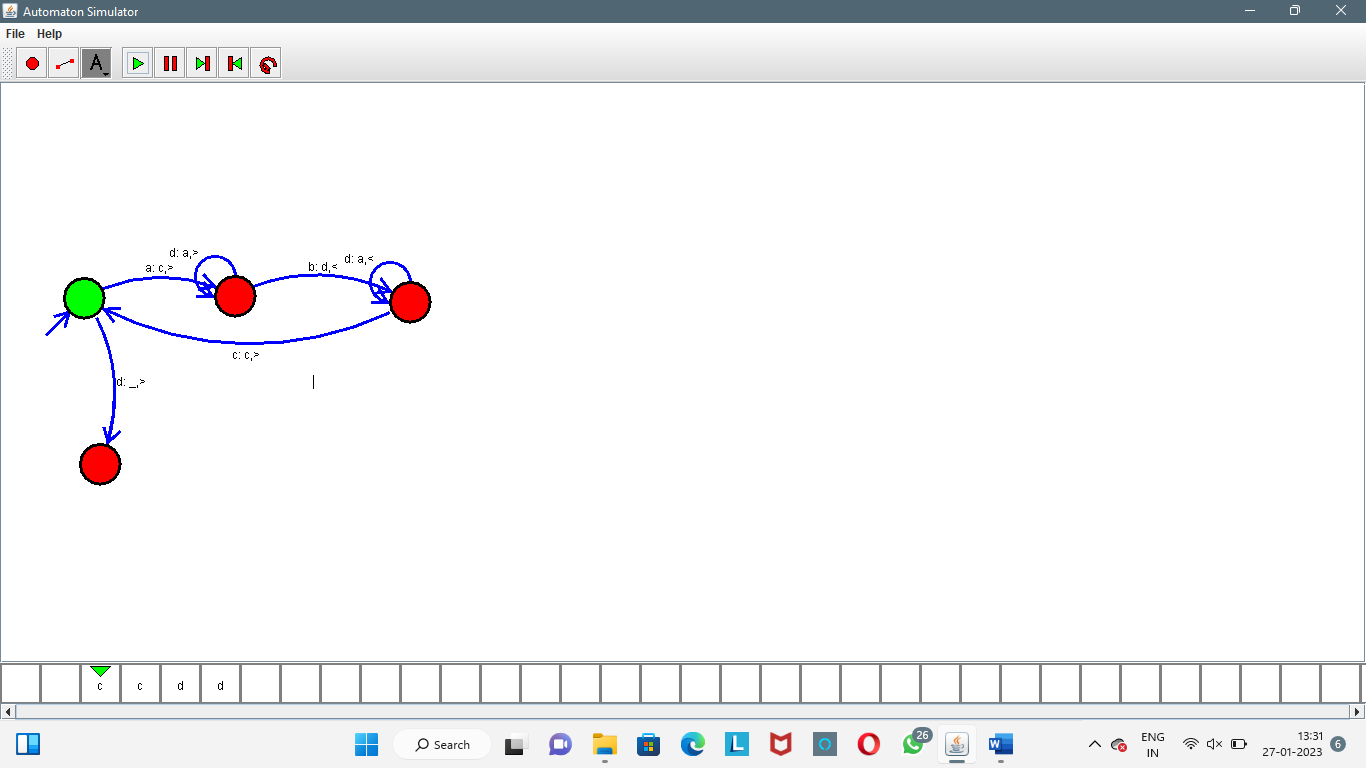
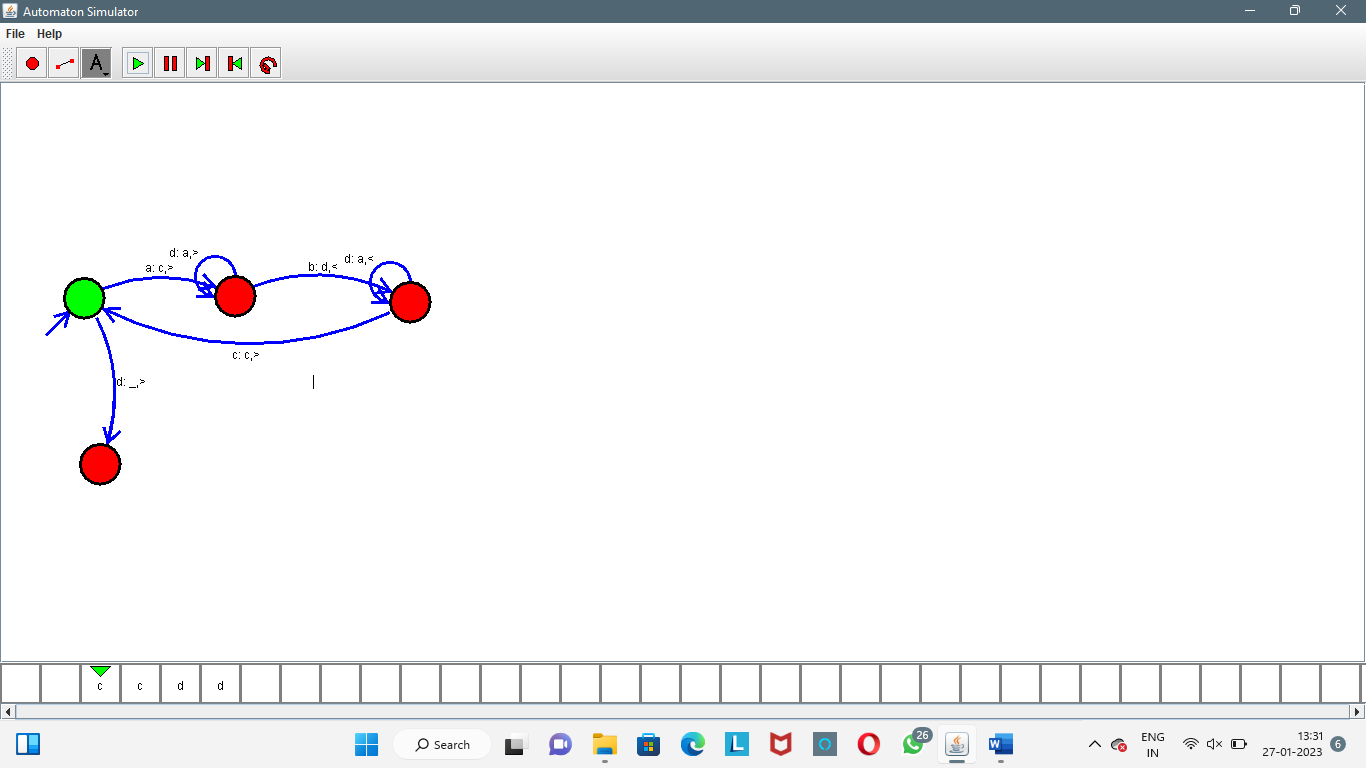
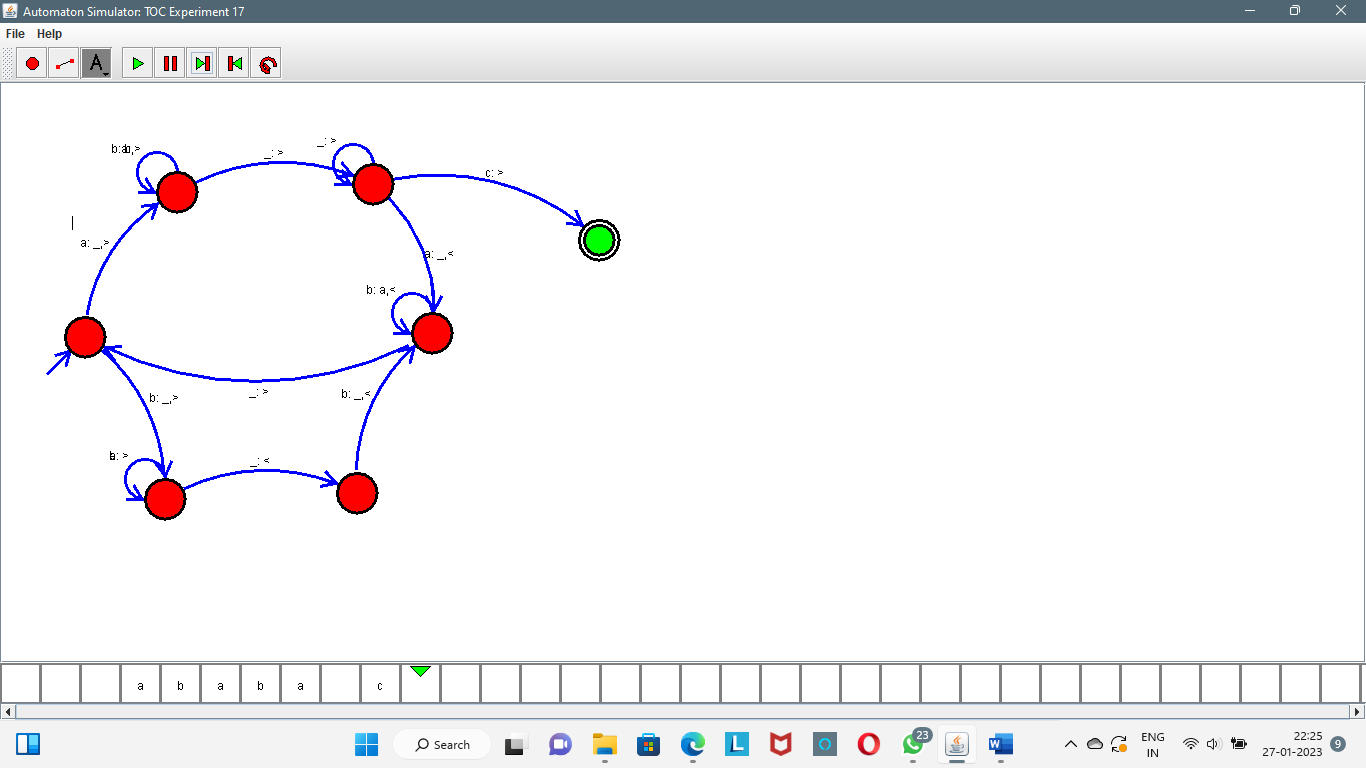
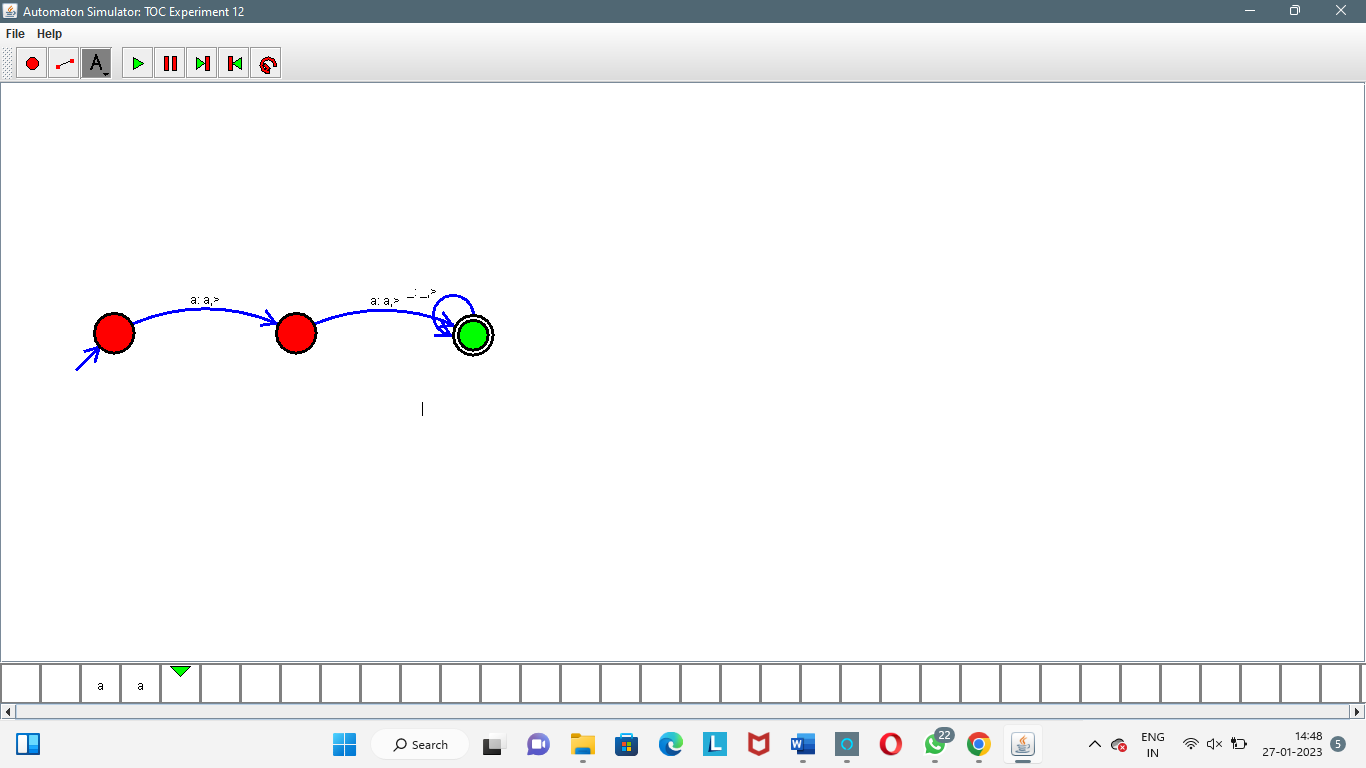
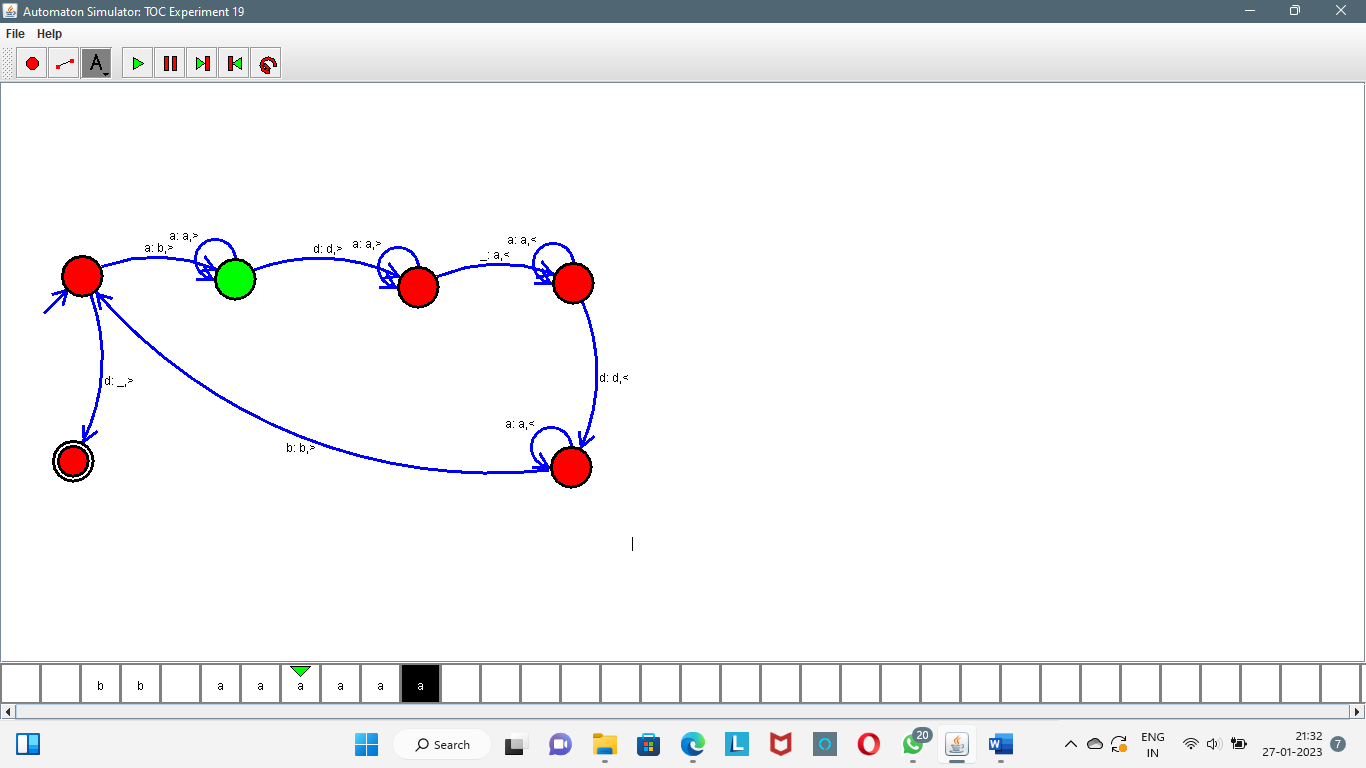
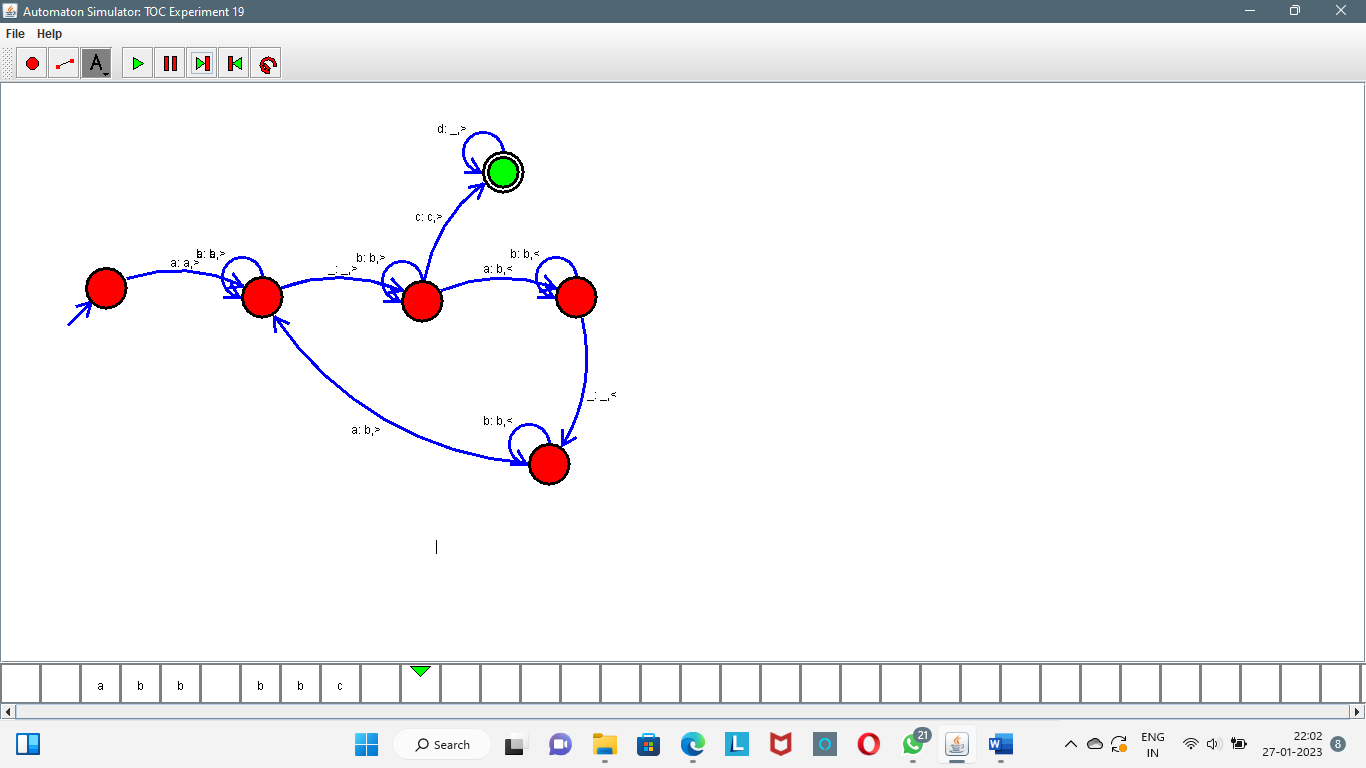
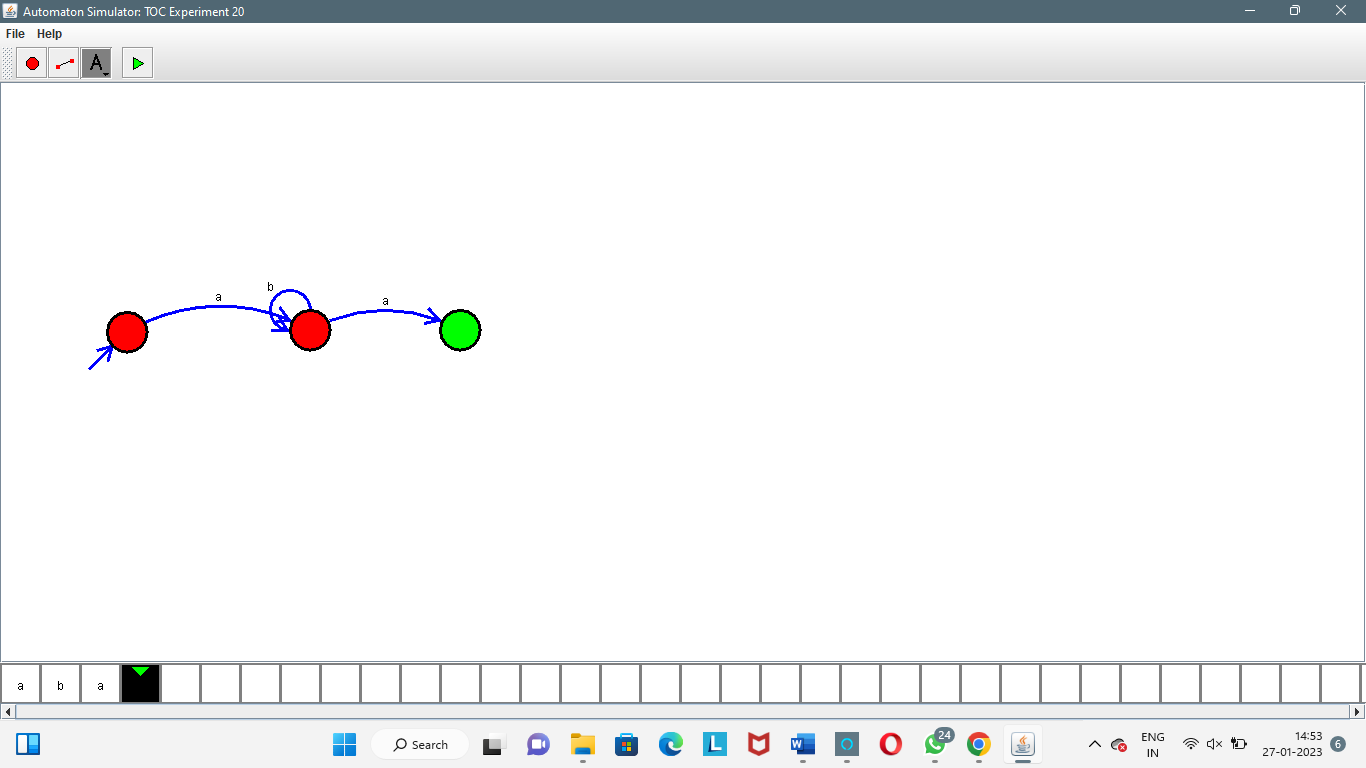
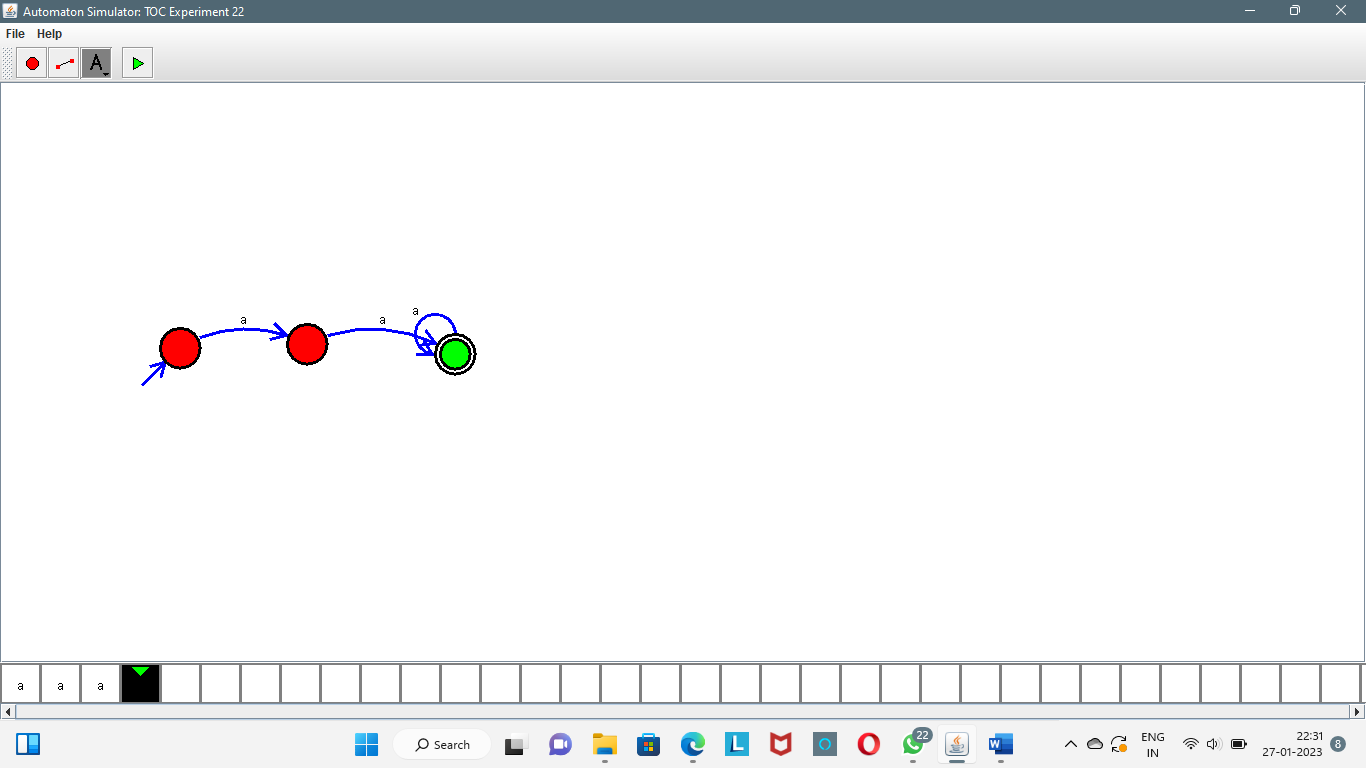
S → 0S0 | A A → 1A | ε

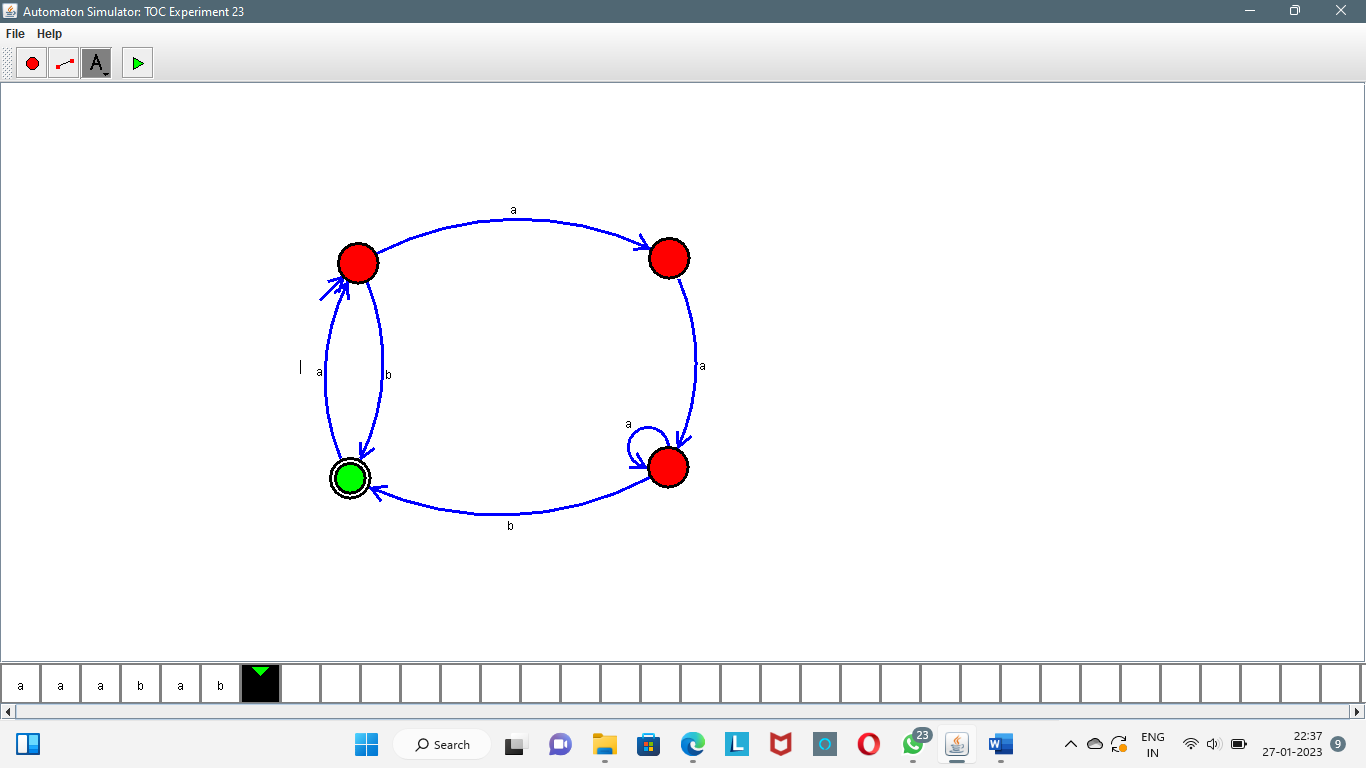
1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

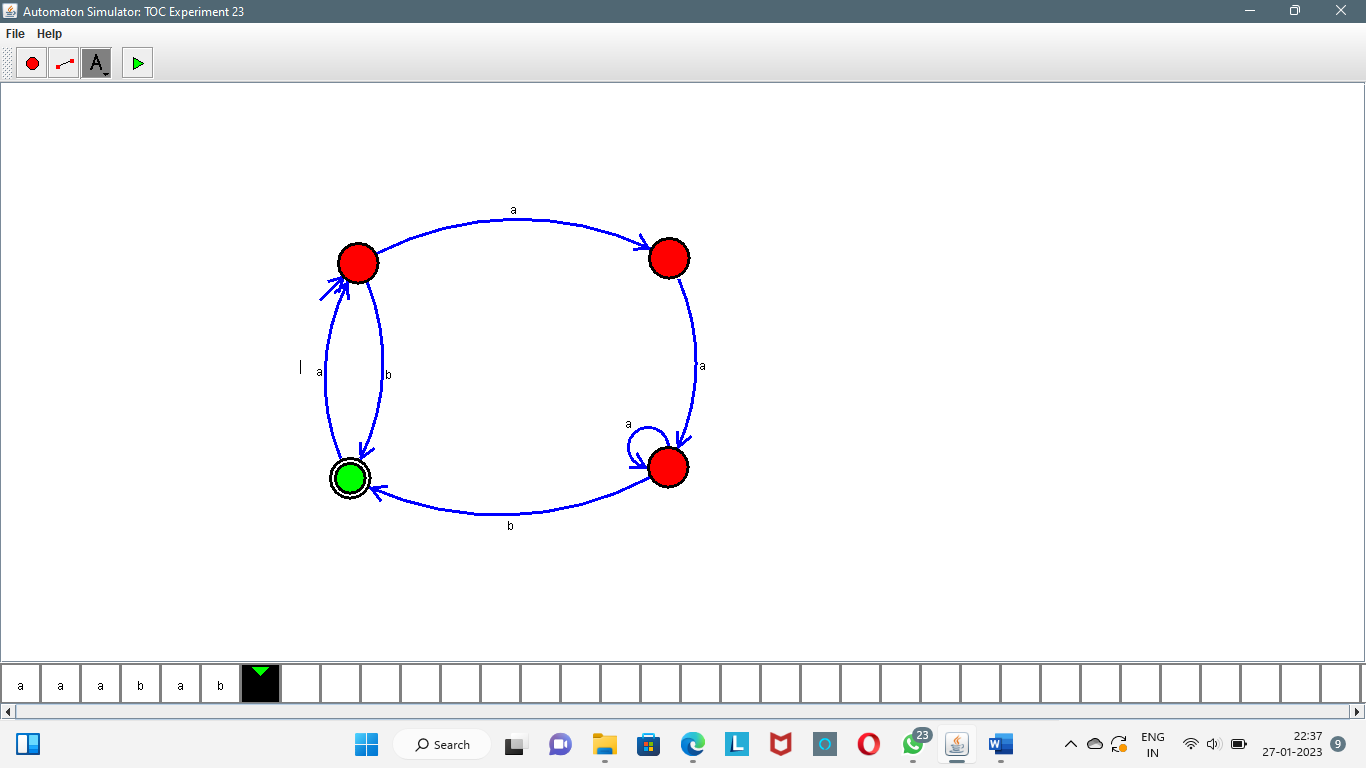
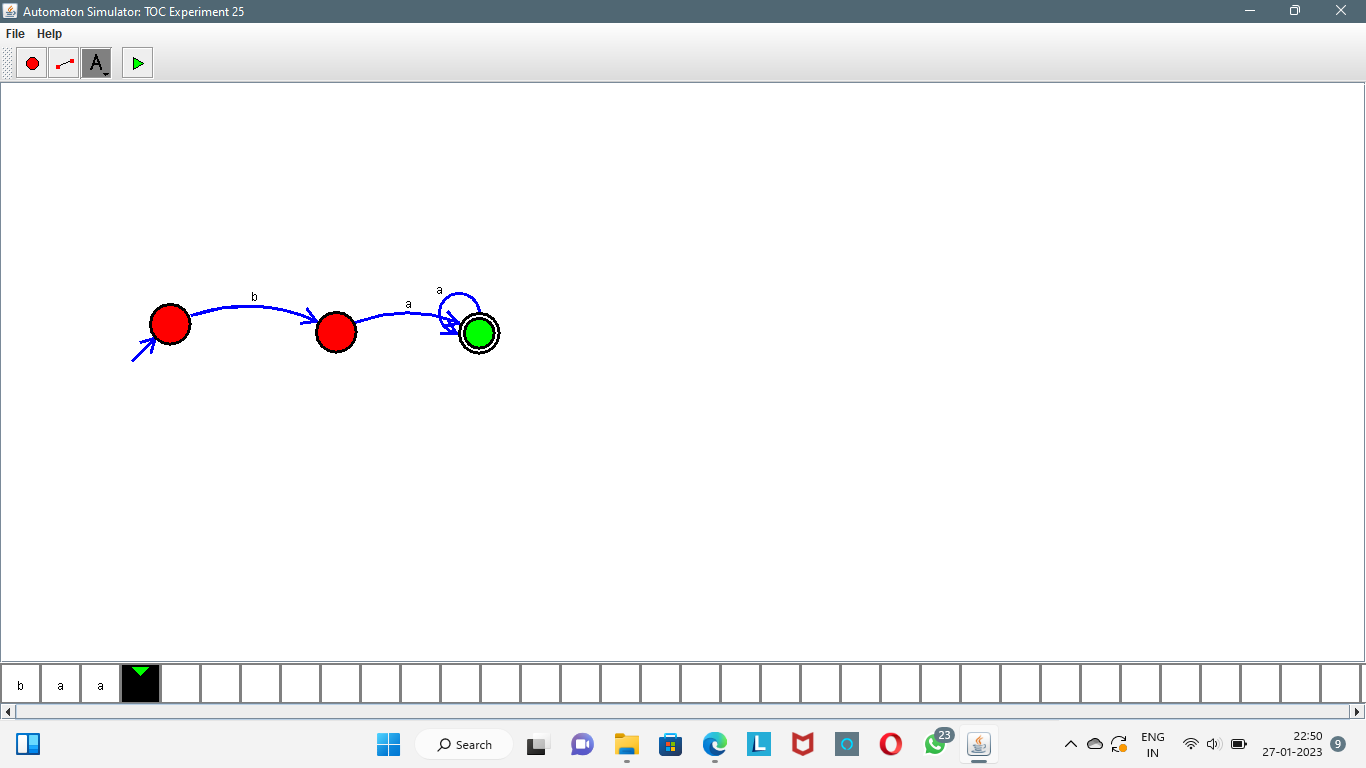
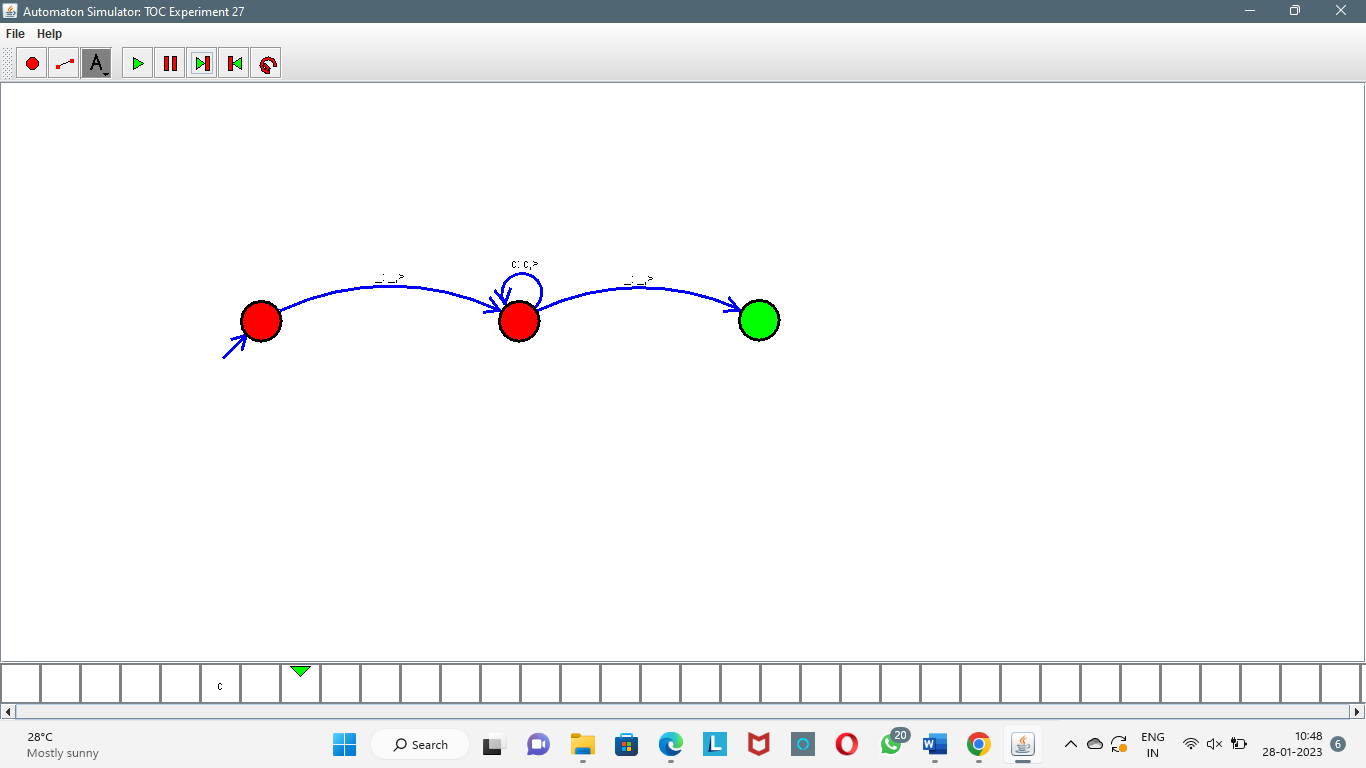
S → 0S1 | ε

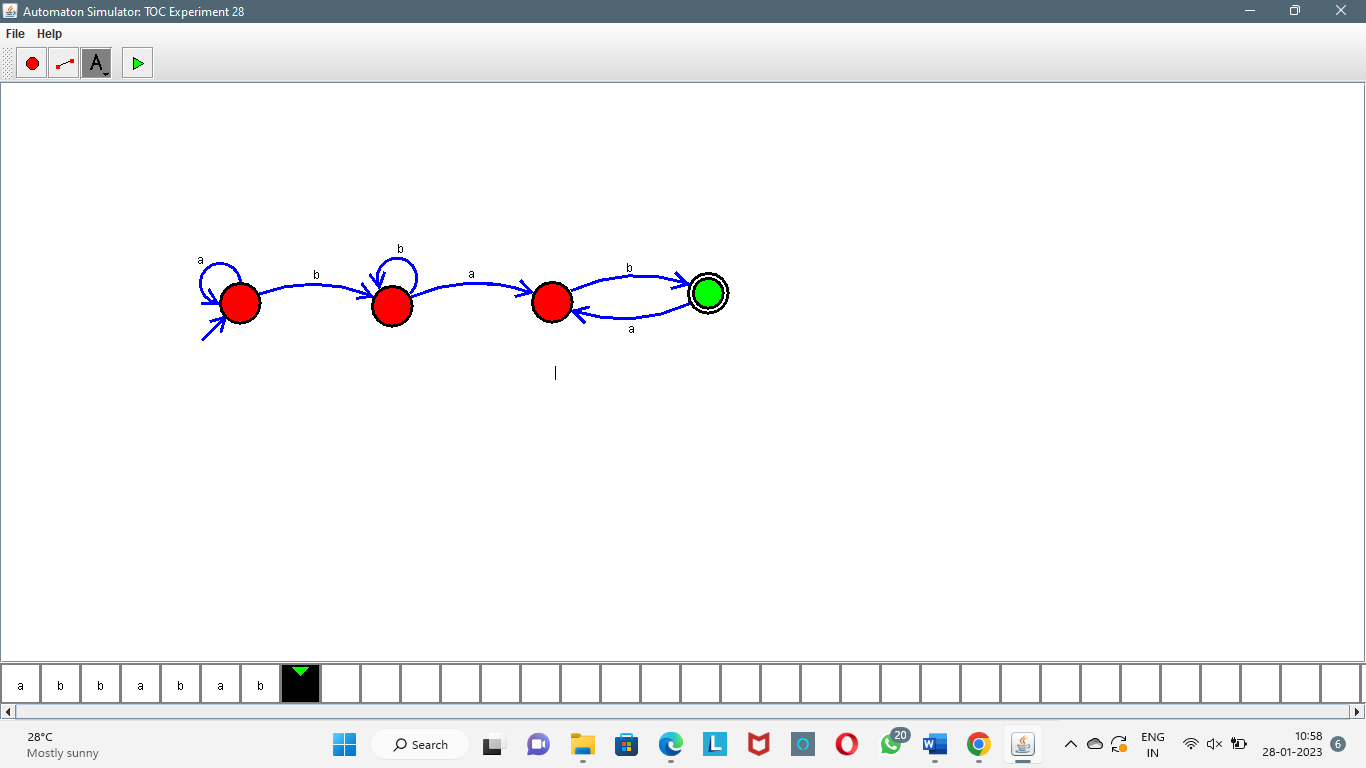
1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

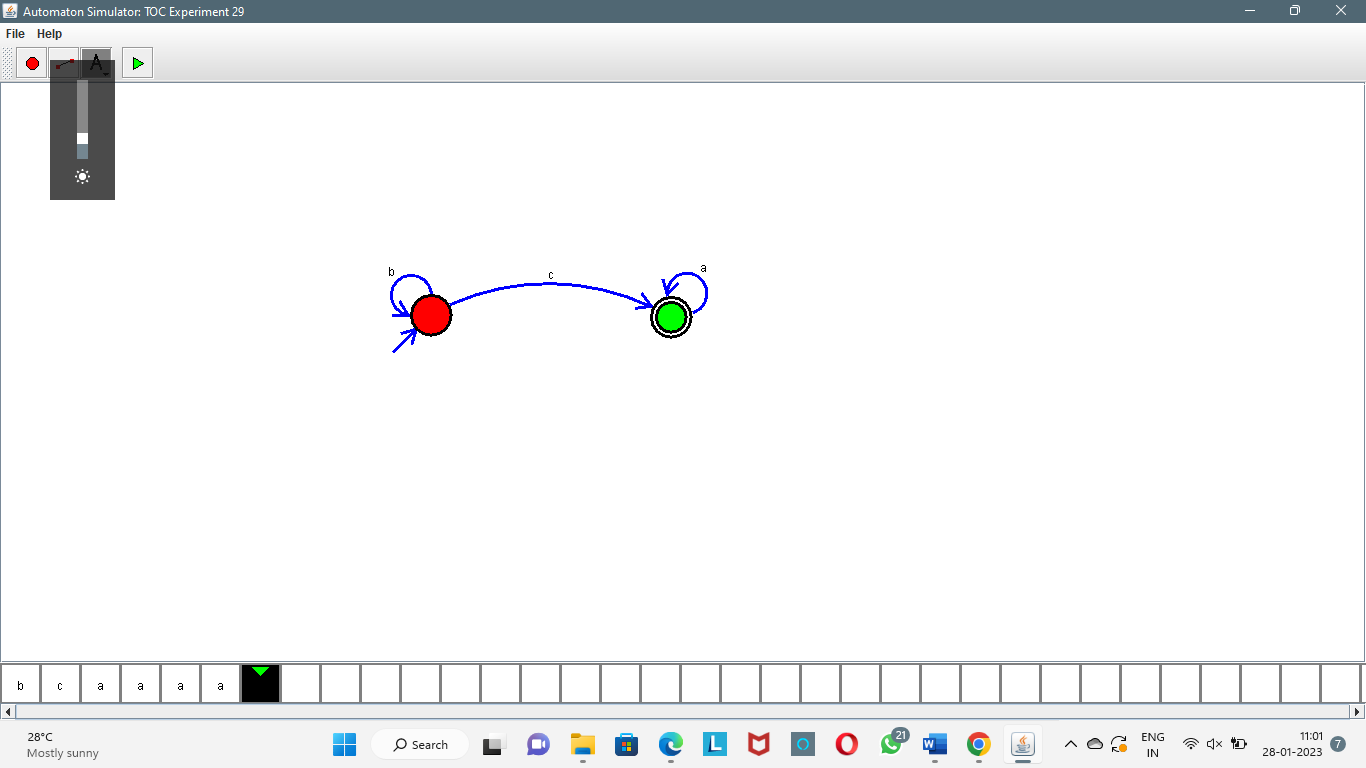
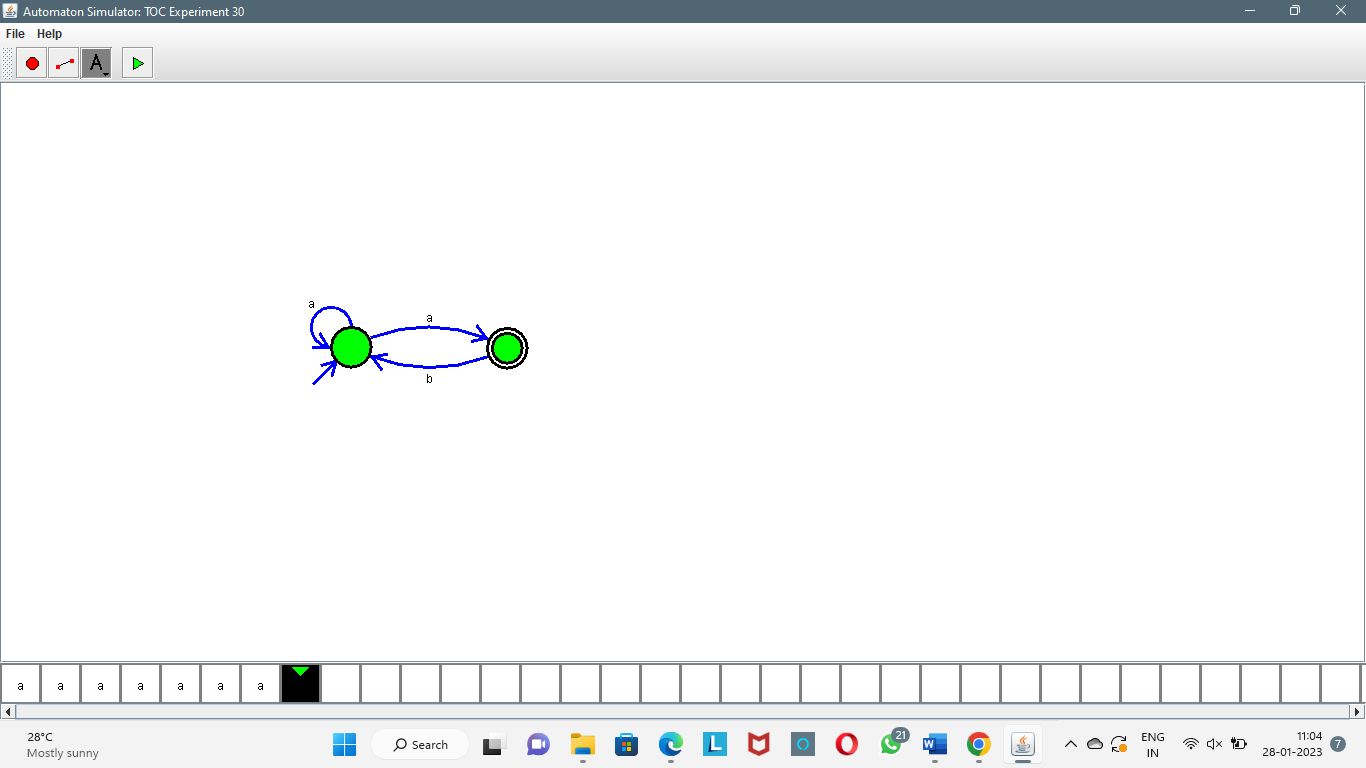
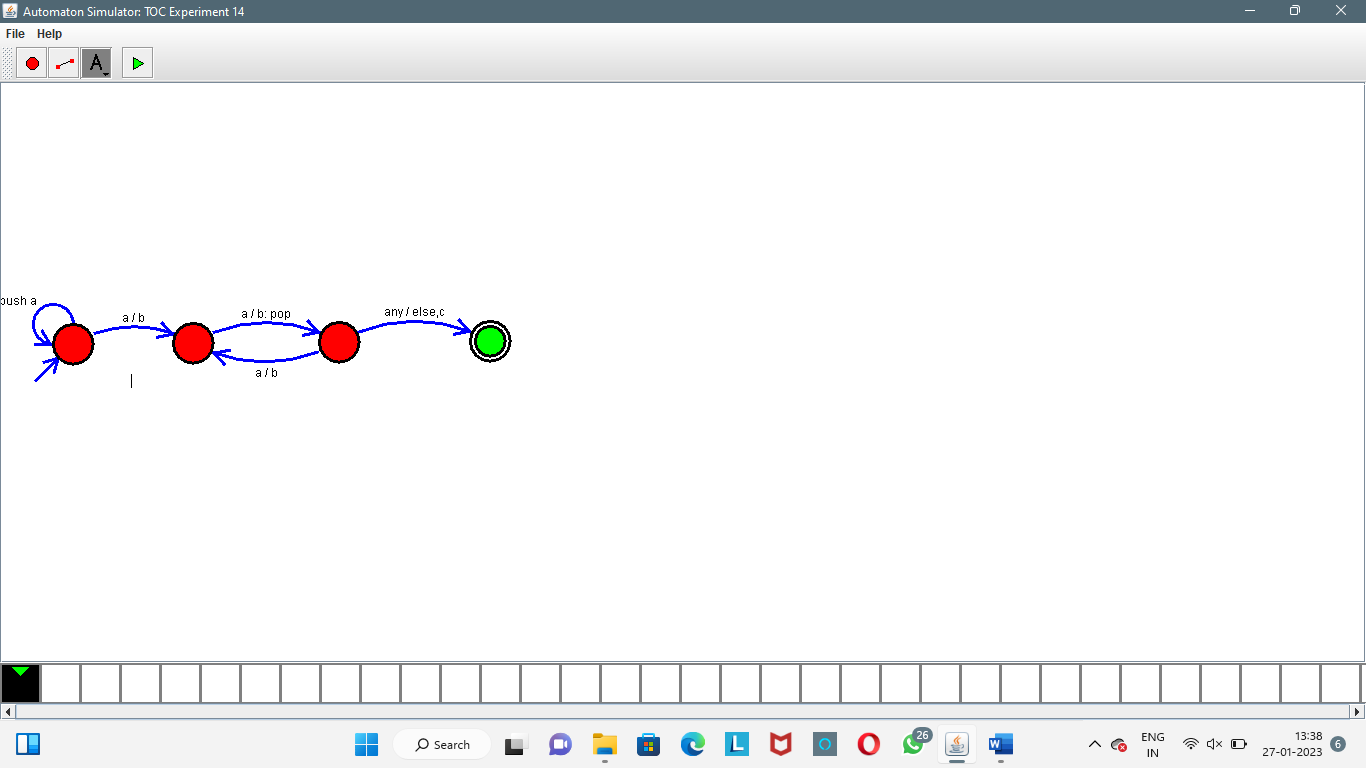
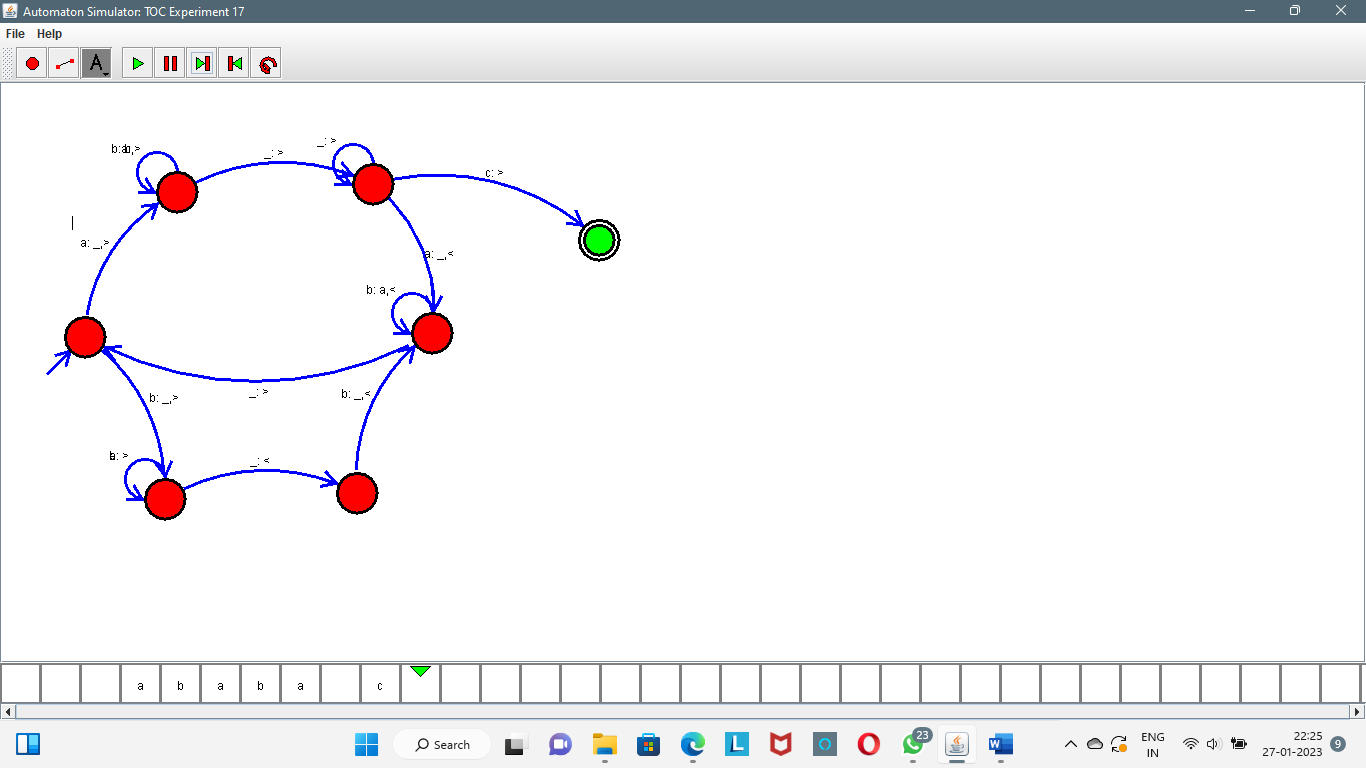
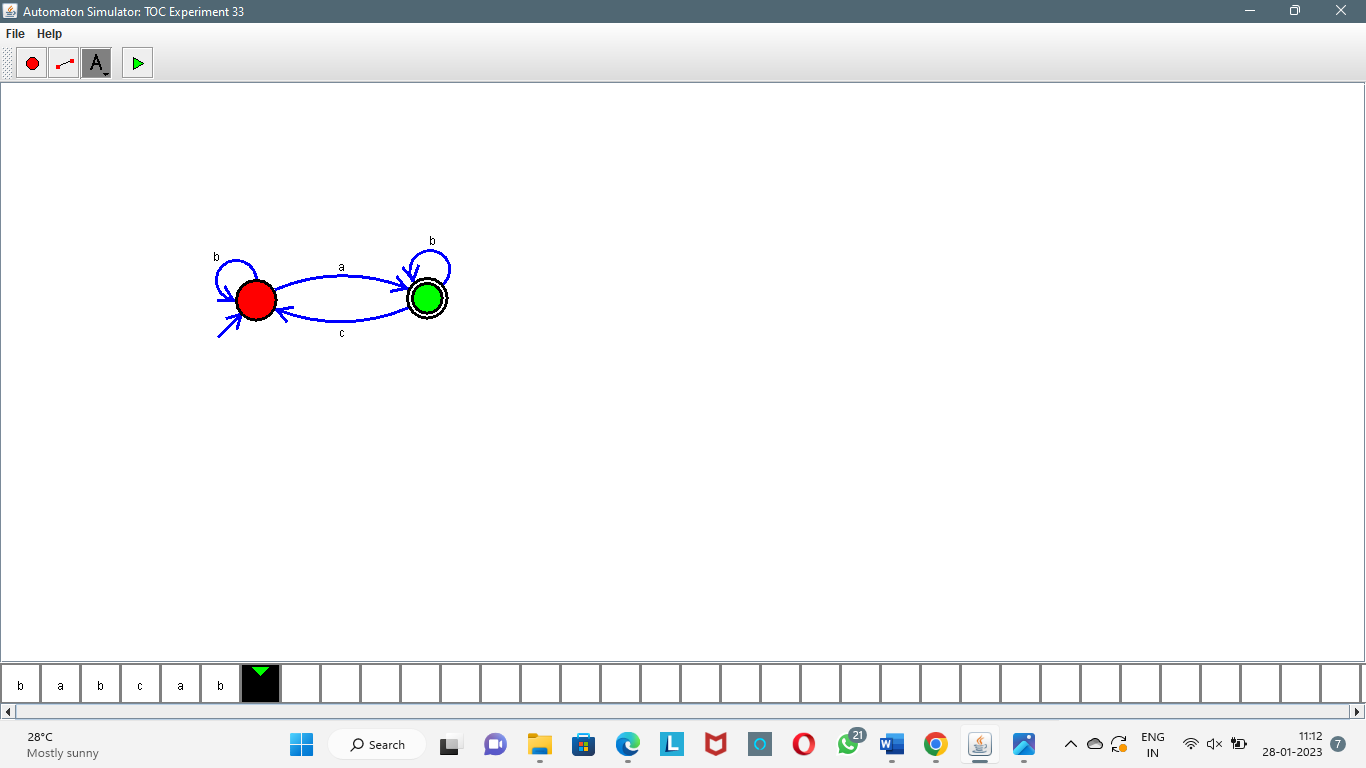
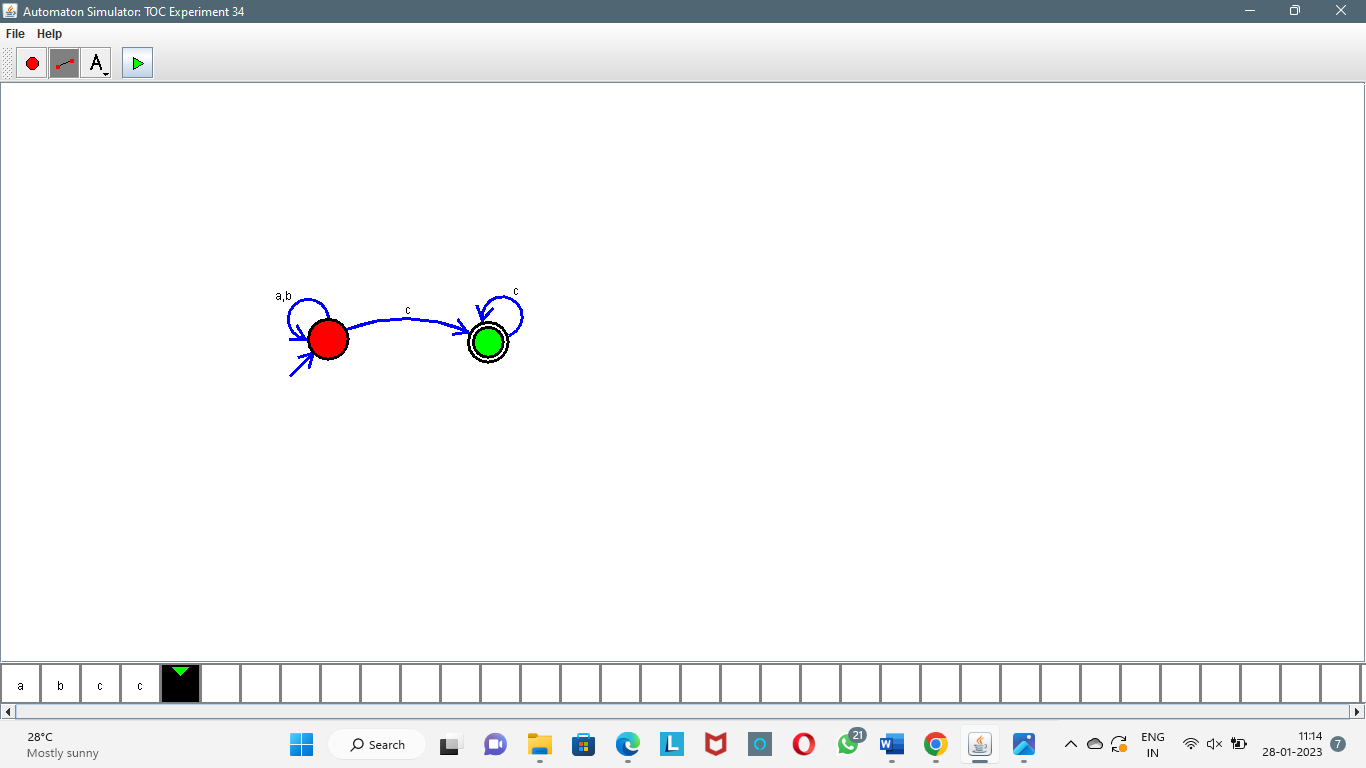
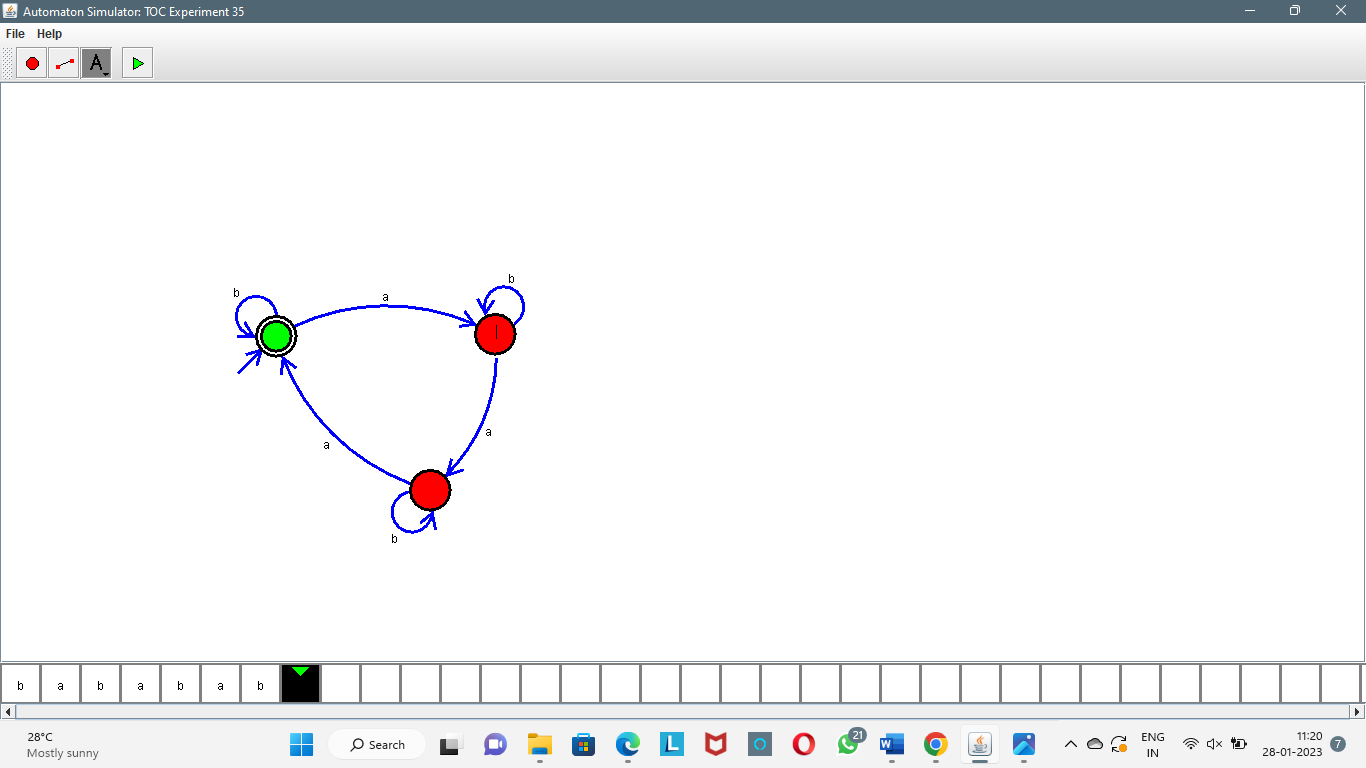
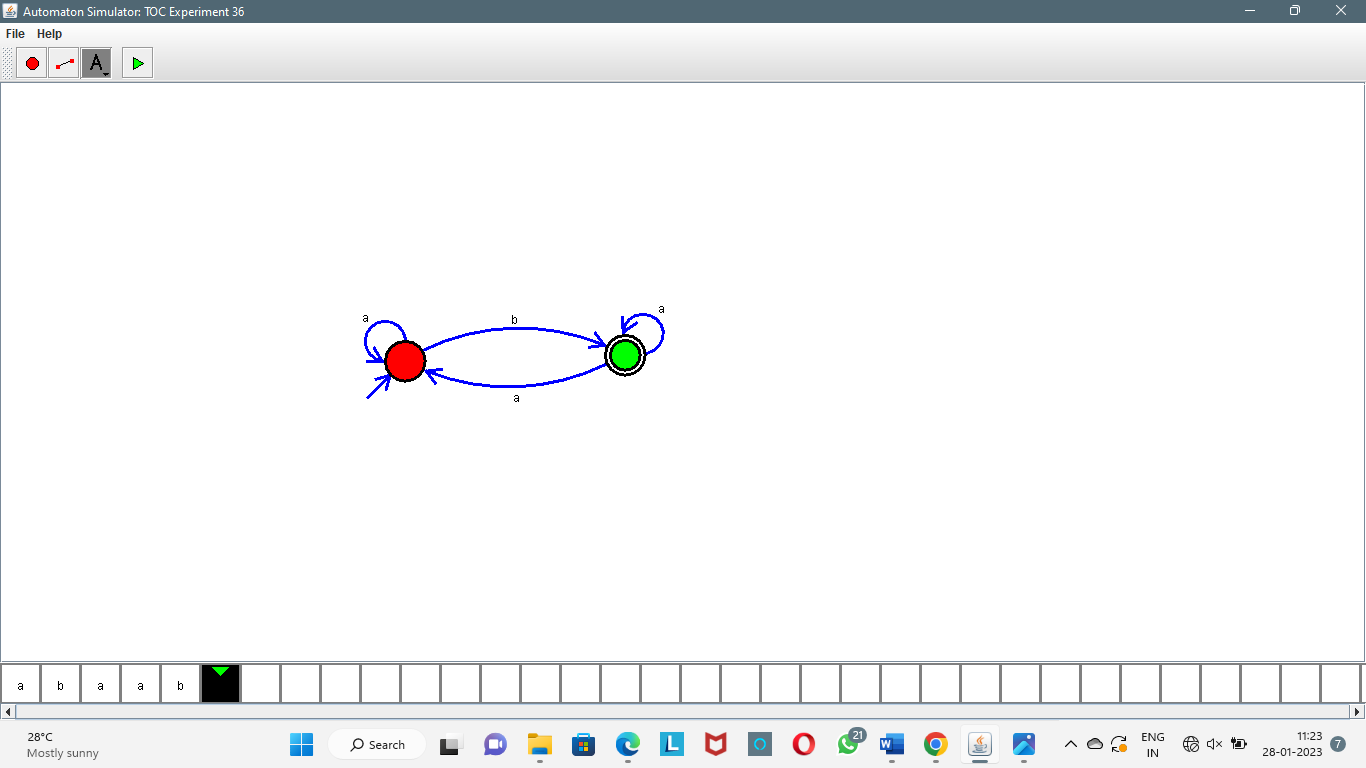
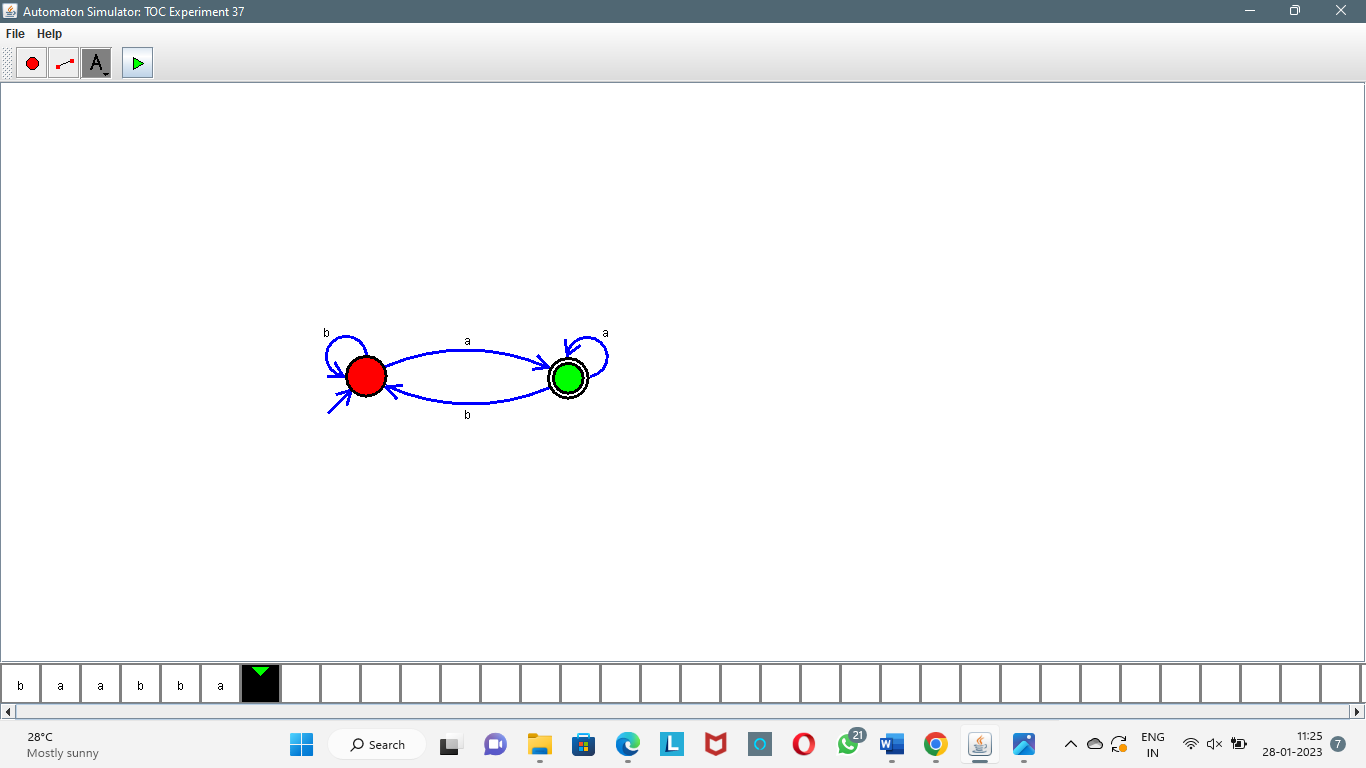
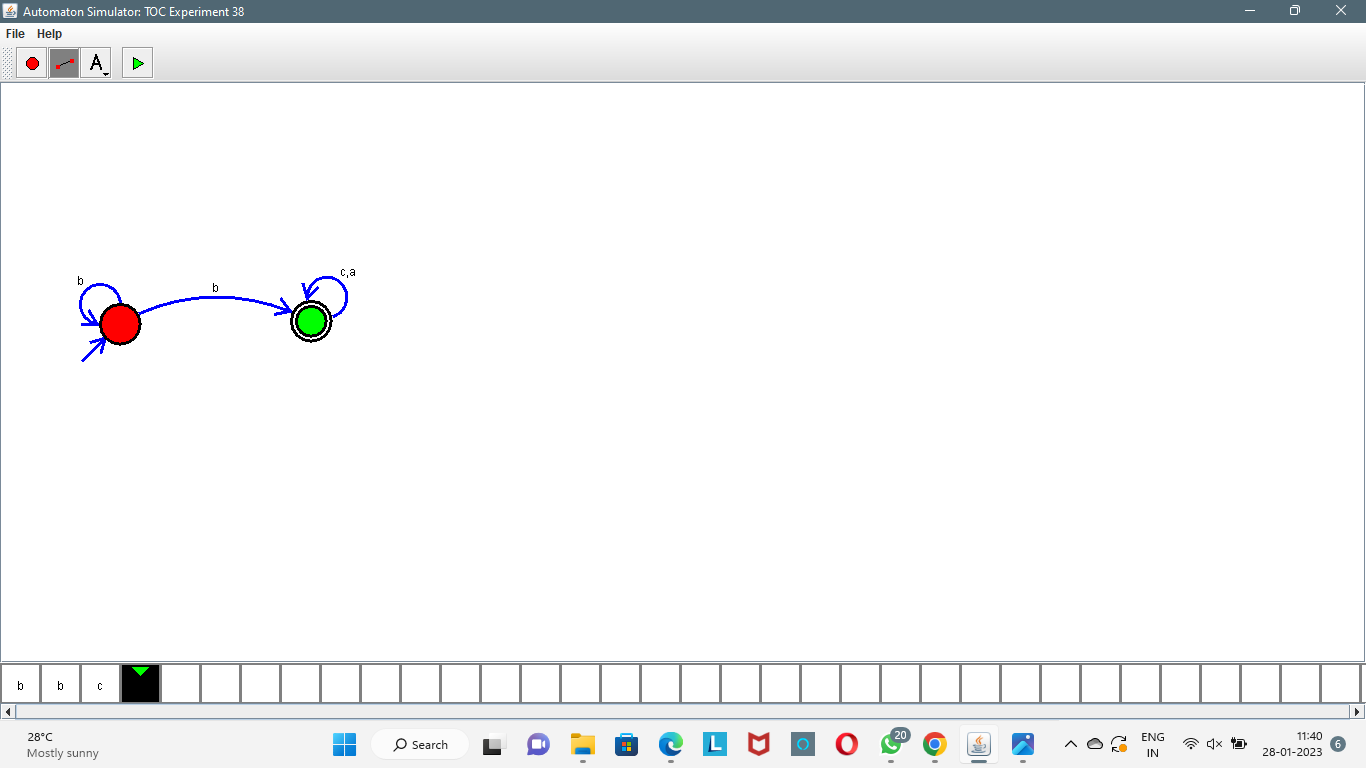
S → A101A, A → 0A | 1A | ε

1. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with b and end with a
2. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with o and end with 1
3. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.
4. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.
5. Design DFA using simulator to accept the input string “a” ,”ac”,and ”bac”. 
6. Design PDA using simulator to accept the input string aabb
7. Design PDA using simulator to accept the input string anb2n
8. Design TM using simulator to accept the input string anbn
9. Design TM using simulator to accept the input string anb2n
10. Design TM using simulator to accept the input string Palindrome ababa
11. Design TM using simulator to accept the input string ww
12. Design TM using simulator to perform addition of ‘aa’ and ‘aaa’
13. Design TM using simulator to perform subtraction of aaa-aa
14. Design DFA using simulator to accept even number of a’s. 
15. Design DFA using simulator to accept odd number of a’s
16. Design DFA using simulator to accept the string the end with ab over set {a,b)

W= aaabab

1. Design DFA using simulator to accept the string having ‘ab’ as substring over the set {a,b}
2. Design DFA using simulator to accept the string start with a or b over the set {a,b}
3. Design TM using simulator to accept the input string Palindrome bbabb
4. Design TM using simulator to accept the input string wcw
5. Design DFA using simulator to accept the string the end with ab over set {a,b)

W= abbaabab

1. Design DFA using simulator to accept the input string “bc” ,”c”,and ”bcaaa”. 
2. Design NFA to accept any number of a’s where input={a,b}.
3. Design PDA using simulator to accept the input string anbn
4. Design TM using simulator to perform string comparison where w={aba aba}
5. Design DFA using simulator to accept the string having ‘abc’ as substring over the set {a,b,c}
6. Design DFA using simulator to accept even number of c’s over the set {a,b,c}
7. Design DFA using simulator to accept strings in which a’s always appear tripled over input {a,b}
8. Design NFA using simulator to accept the string the start with a and end with b over set {a,b} and check W= abaab is accepted or not. 
9. Design NFA using simulator to accept the string that start and end with different symbols over the input {a,b}.
10. Design NFA using simulator to accept the input string “bbc” ,”c”,and ”bcaaa”. 
11. Design DFA using simulator to accept the string the end with abc over set {a,b,c)

W= abbaababc

1. Design NFA to accept any number of b’s where input={a,b}.