**Programs**

**[8] NFA Which Accepts Strings Start With ‘B’ And End With ‘A’**

#include <stdio.h>

#include <stdbool.h>

int transitions[][3] = {

{1, -1, -1}, {1, 2, -1}, {1, -1, 3}, {3, -1, 3}

};

bool simulateNFA(const char \*input)

{

int currentState = 0;

for (int i = 0; input[i] != '\0'; i++)

{

char symbol = input[i];

int symbolIndex = -1;

if (symbol == 'b')

symbolIndex = 0;

else if (symbol == 'a')

symbolIndex = 1;

else

return false;

int nextState1 = transitions[currentState][symbolIndex];

int nextState2 = transitions[currentState][2];

if (nextState1 == -1 && nextState2 == -1)

return false;

if (nextState1 != -1)

currentState = nextState1;

else

currentState = nextState2;

}

return currentState == 3;

}

int main()

{

char input[100];

printf("Enter a String : ");

scanf("%s", input);

if (simulateNFA(input))

printf("Accepted\n");

else

printf("Rejected\n");

return 0;

}

**Algorithm**

Input\_String: The string to be checked by the NFA

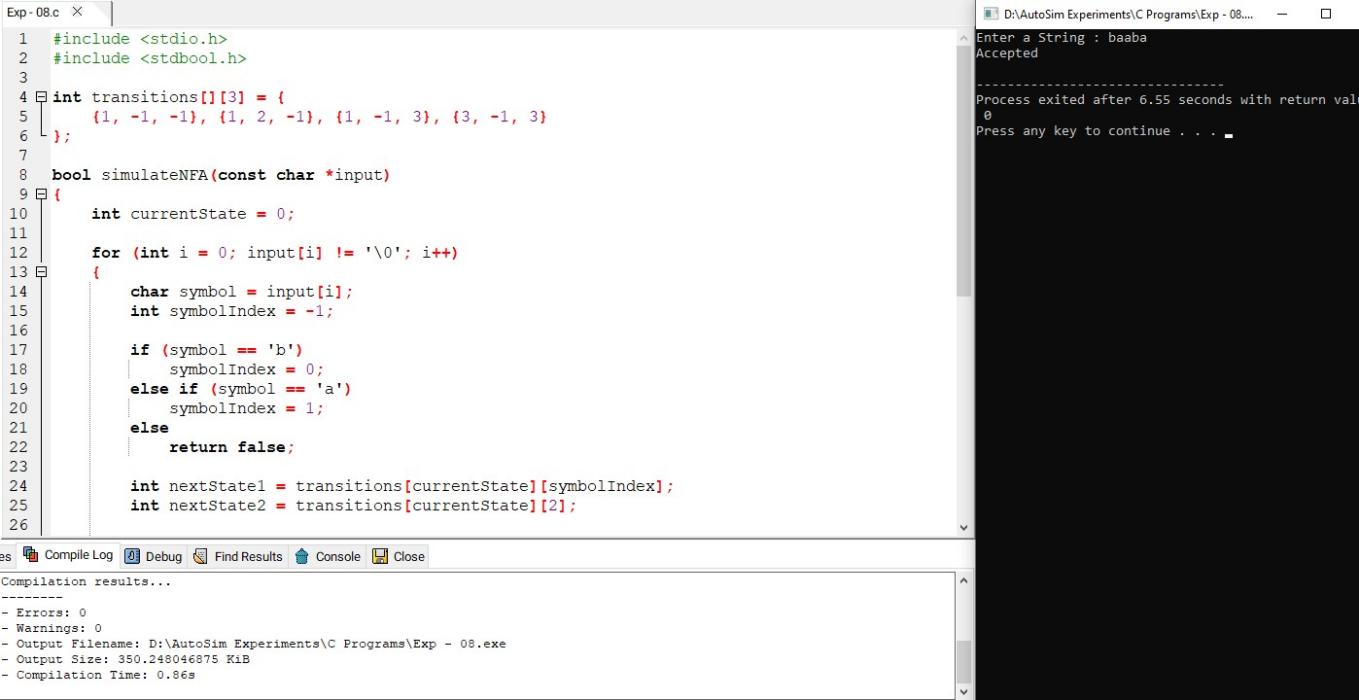
Initialize the current state to the initial state of the NFA.

For each character (symbol) c in Input\_String

a. Get the set of next states from the current state using the transition function for symbol c. b. Update the current state to the union of the next states.

After processing all characters in Input\_String, check if the current state contains the accepting state (i.e., if it is possible to reach the accepting state from the current state).

If the current state contains the accepting state, output "Accepted," indicating that the input string satisfies the condition (starts with 'B' and ends with 'A'). Otherwise, output "Rejected."



**Result**

Thus we have successfully implemented and executed the program and the strings given as inputs are verified