DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
RAMAIAH INSTITUTE OF TECHNOLOGY
(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)
M. S. R. I. T. POST, BANGALORE – 560054

A Report On JFLAP Tool

for the subject

FINITE AUTOMATA AND FORMAL LANGUAGE (IS44)

In

Fourth Semester

Submitted By,

Nivedita G S	**1MS20IS080**
Owais Iqbal	**1MS20IS081**

Submitted to,

Dr. S R Mani Shekar

Assistant Professor

Dept. of ISE, RIT

TABLE OF CONTENTS

PROBLEM STATEMENT

Design a PDA to accept the language

L ={w| na(w) = nb(w)} by final state.

Explanation:

The language accepted by the machine should consist strings of a’s and b’s of any length. Only restriction is that number of a’s in string w should be equal to number of b’s. The order of a’s and b’s is irrelevant.

The strings that are accepted by the language are :

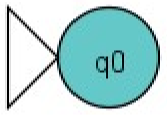
ε, ab, ba, aabb, aaabbb, ababab, aabbabab ……

The strings that are not accepted by the language are :

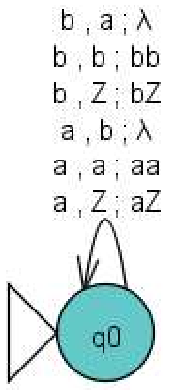
a, b, aab, bbaaa, abababb, ……

SOLUTION TO THE PROBLEM

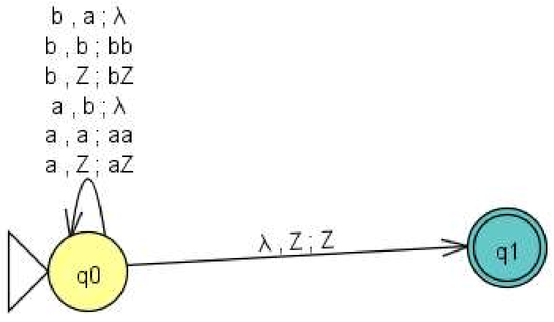
Initial State:



Next State:



Final State:



COMPONENTS

$M = \{ Q, \Sigma, \Gamma, \delta, q0, Z0, F\}$

Where, Q is set of states

Σ is the set of input characters

Γ is the stack top symbols

δ is the transition function

q0 is the start state

z0 is the stack top symbol

F is the final state

$Q = \{ q0, q1 \}$

$\Sigma = \{ a, b\}$

$\Gamma = \{ Z0, a, b \}$

$q0 = \{ q0 \}$

$Z0 = \{ Z0 \}$

$F = \{ q1 \}$

$\delta =$ $\delta(q0, a, Z0) = (q0, aZ0)$

$\delta(q0, b, Z0) = (q0, bZ0)$

$\delta(q0, a, a) = (q0, aa)$

$\delta(q0, b, b) = (q0, bb)$

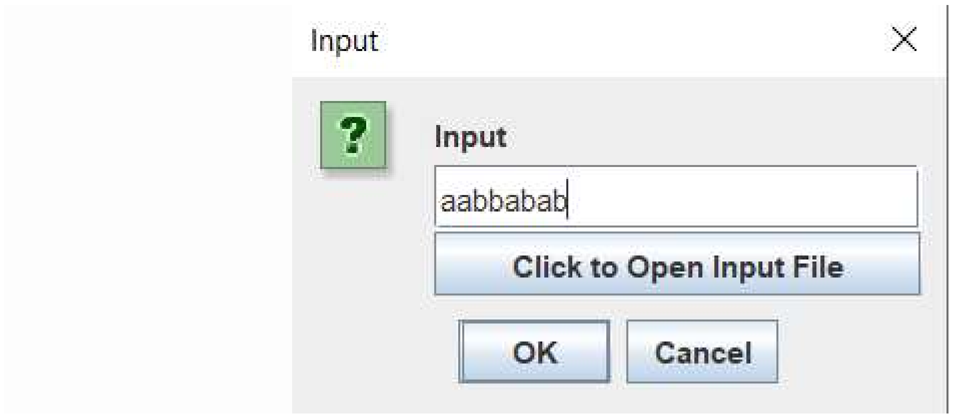
$\delta(q0, a, b) = (q0, \varepsilon)$

$\delta(q0, b, a) = (q0, \varepsilon)$

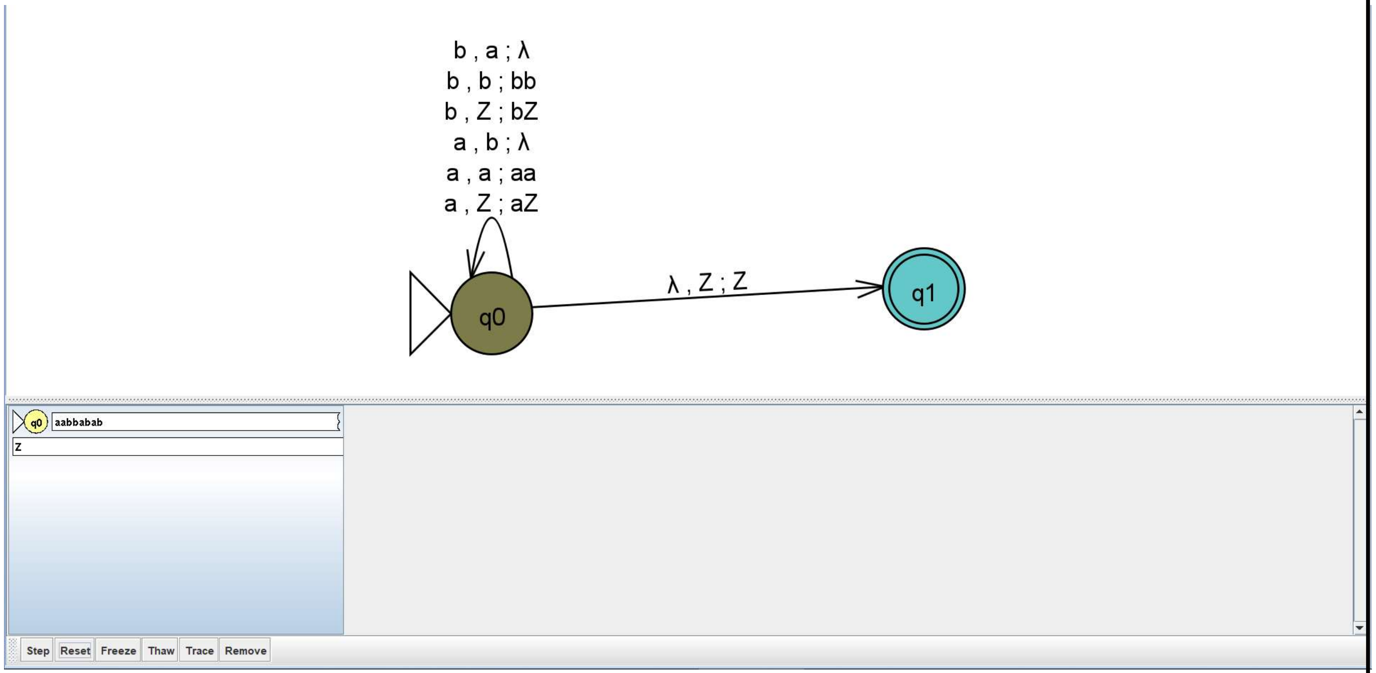
$\delta(q0, \varepsilon, Z0) = (q1, Z0)$

EXECUTION

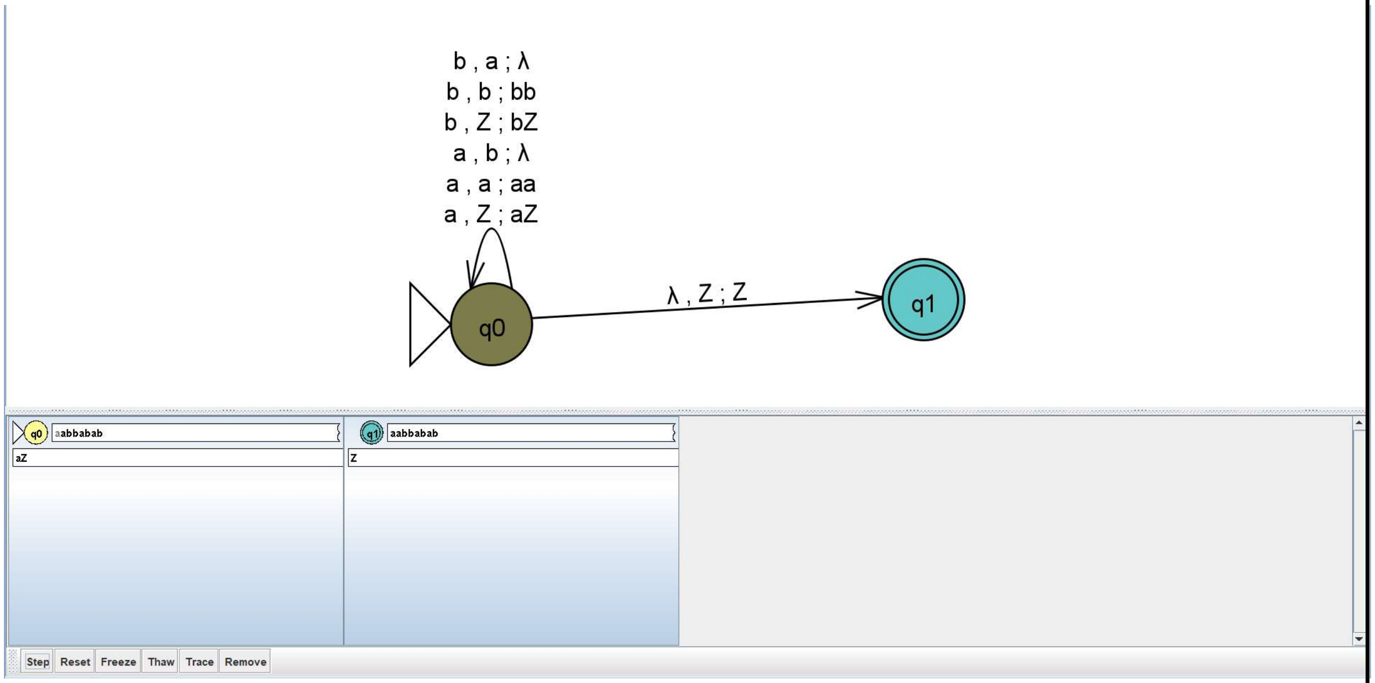
Input:



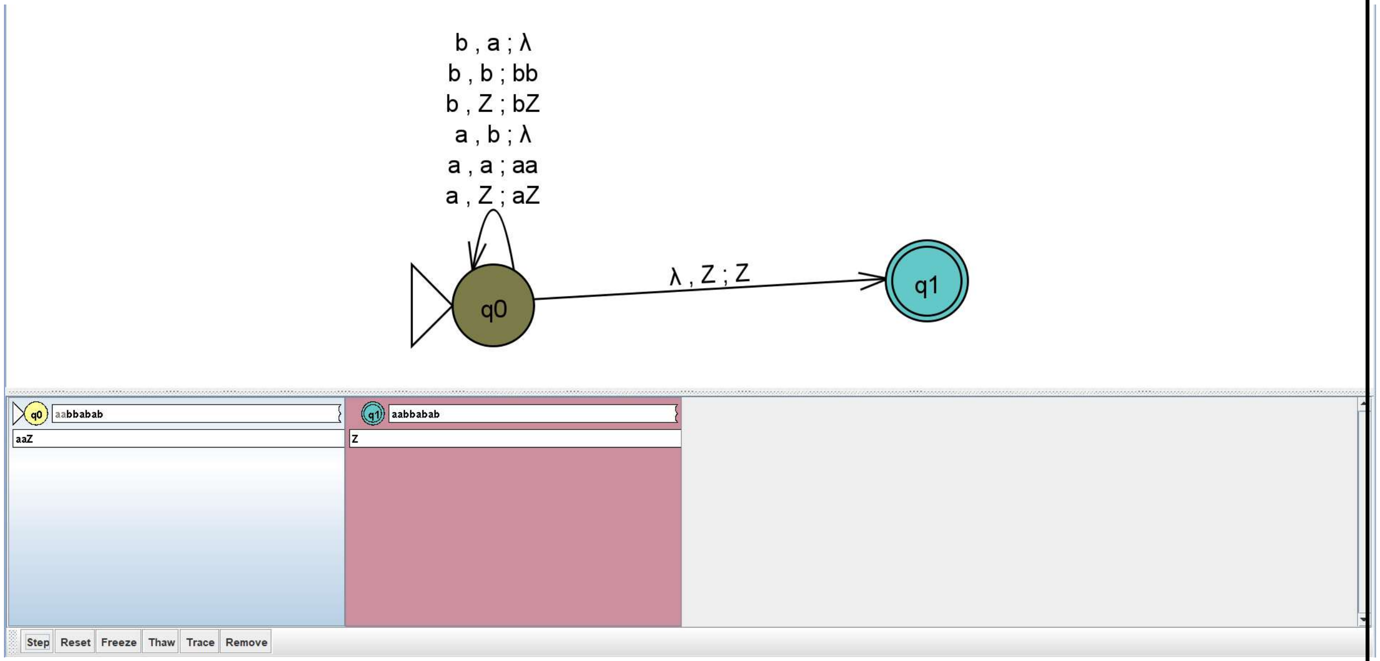
Here we have given input as aabbabab.



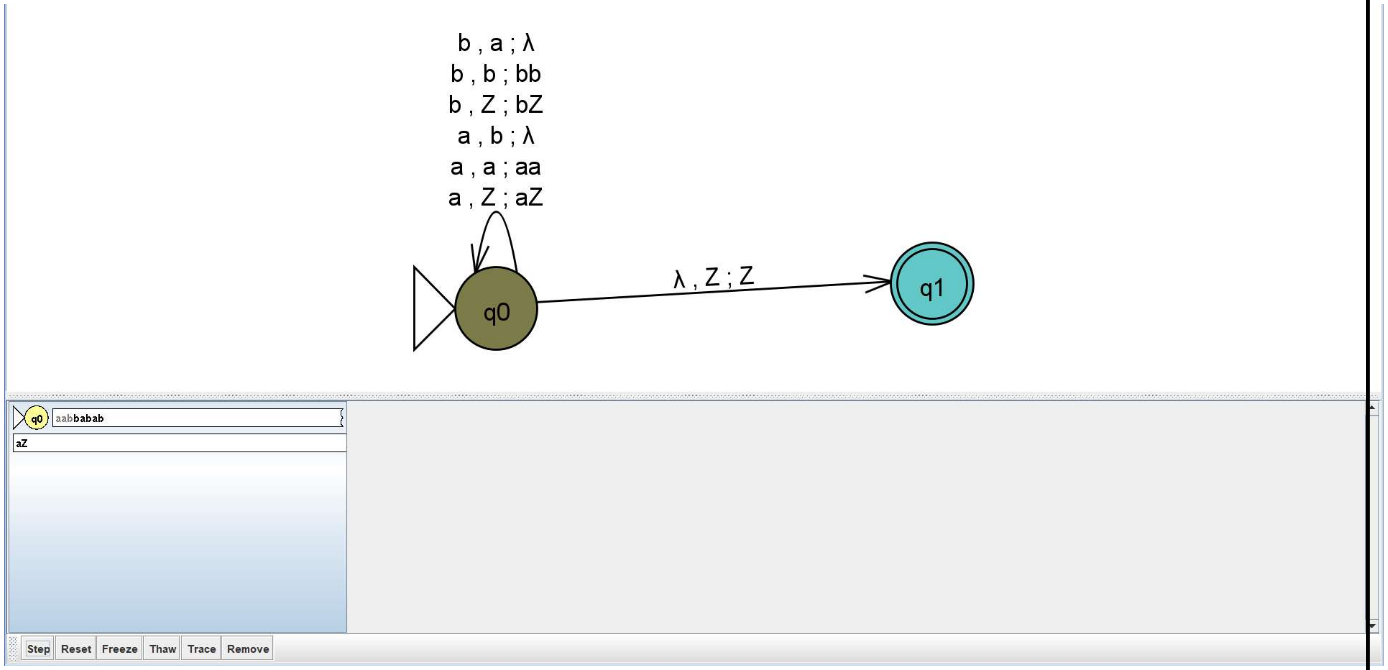
Initial state, in state q0, Z is the stack top



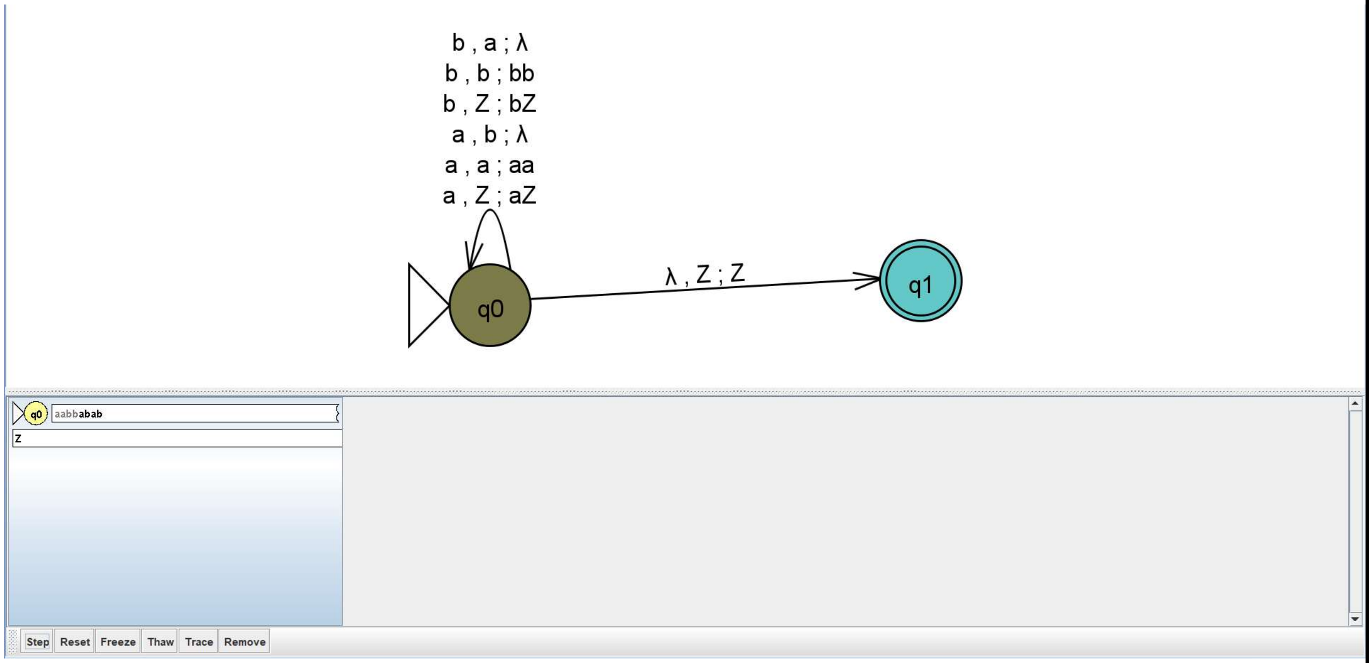
Here, input is ‘a’, and stack top is Z so it pushes ‘a’ into the stack.



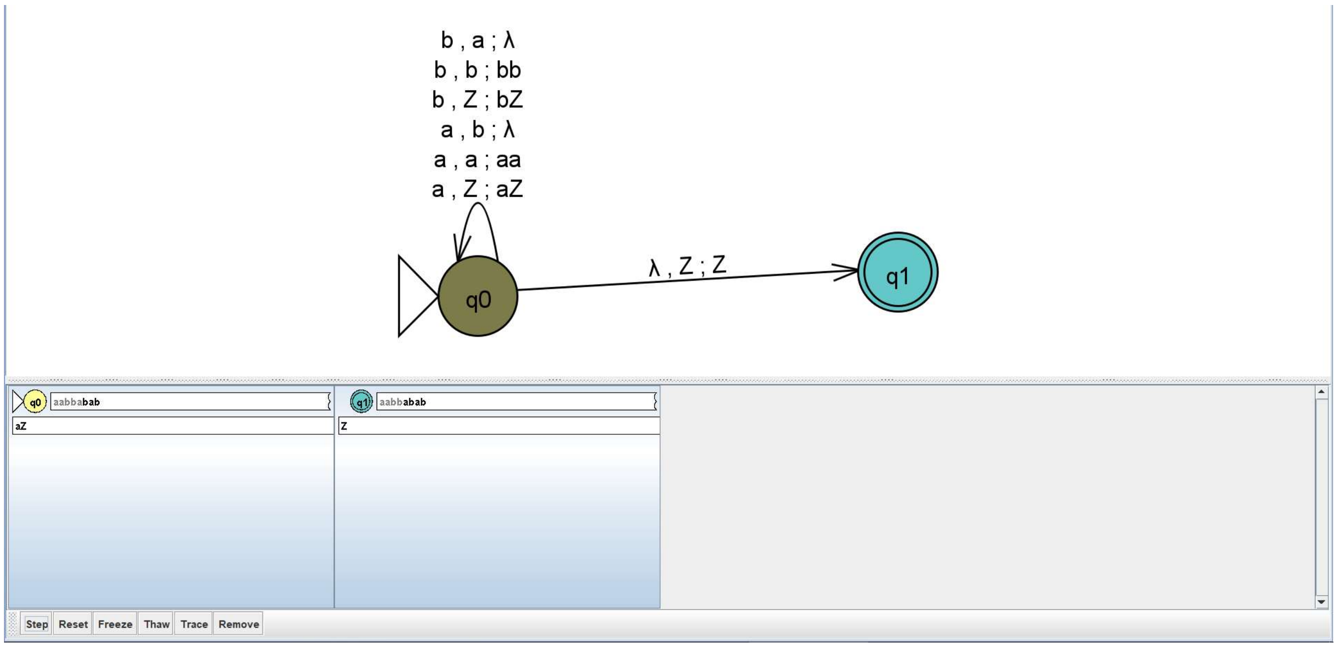
Here, the input is ‘a’, and stack top is ‘a’ so it pushes ‘a’ into the stack.



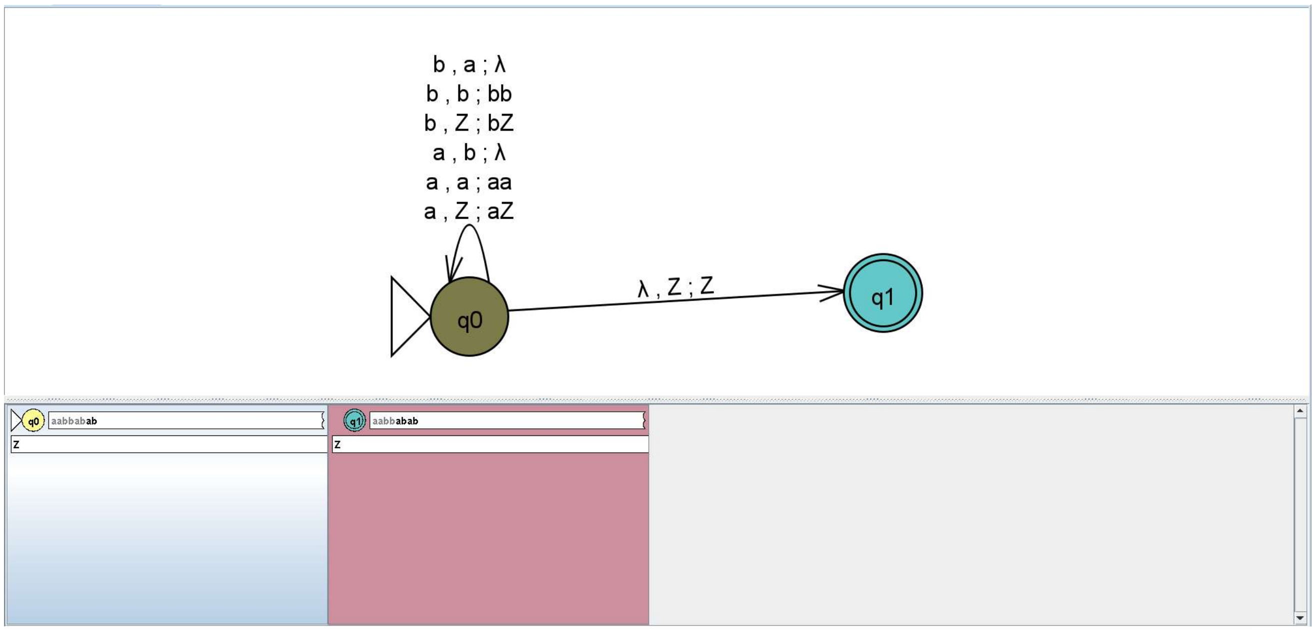
Here the input is ‘b’ and stack top is ‘a’, so it pops out one ‘a’ from the stack.



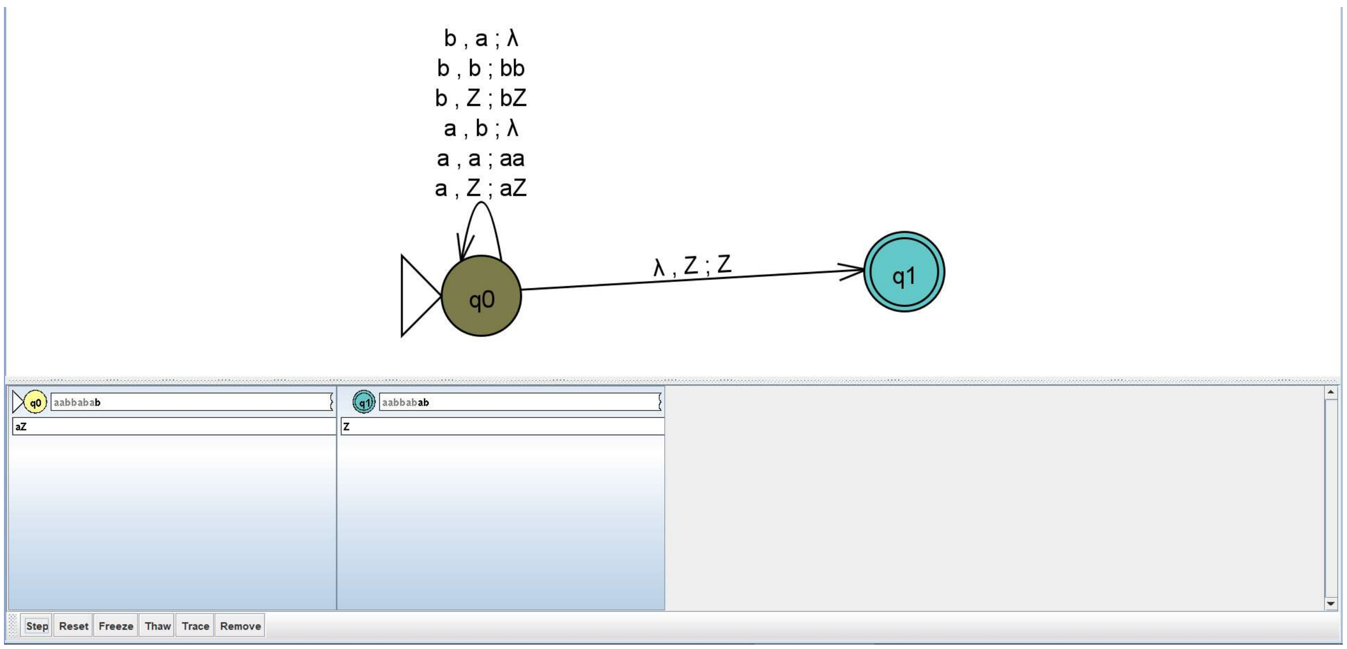
Here the input is ‘b’ and stack top is ‘a’, so it pops out one ‘a’ from the stack.



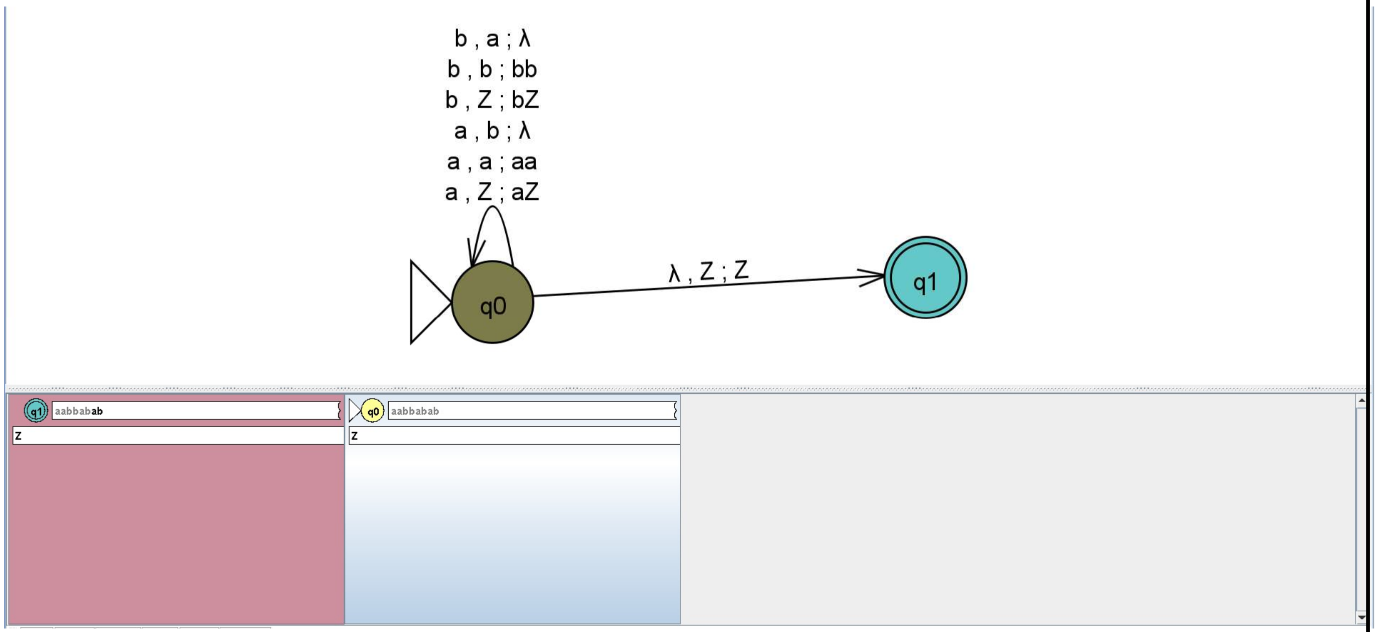
Here, input is ‘a’, and stack top is Z so it pushes ‘a’ into the stack.



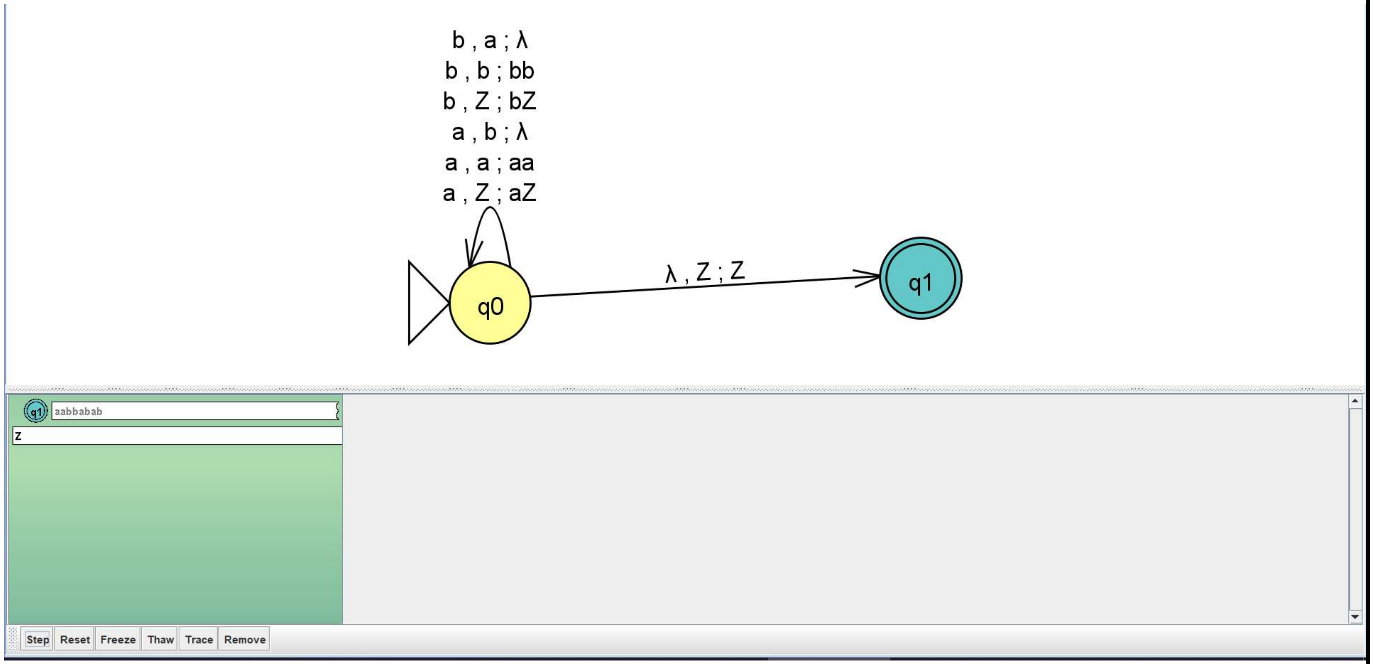
Here the input is ‘b’ and stack top is ‘a’, so it pops out one ‘a’ from the stack.



Here, input is ‘a’, and stack top is Z so it pushes ‘a’ into the stack.



Here the input is ‘b’ and stack top is ‘a’, so it pops out one ‘a’ from the stack and goes to the next state q1.



Here, there is no input i.e., epsilon input and stack top is Z so it goes to the final state.

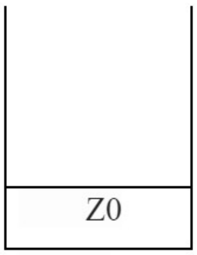
TRACING

Instantaneous description

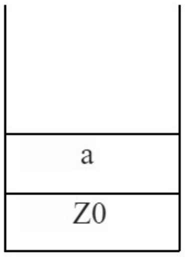
String: aabbabab

```
(q0, aabbabab, Z0)  |-  (q0, abbabab, aZ0)
                    |-  (q0, bbabab, aaZ0)
                    |-  (q0, babab, aZ0)
                    |-  (q0, abab, Z0)
                    |-  (q0, bab, aZ0)
                    |-  (q0, ab, Z0)
                    |-  (q0, b, az0)
                    |-  (q0,  ε, Z0)
                    |-  (q1, ε, Z0)
```

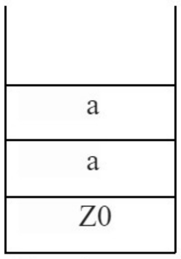
Stack tracing



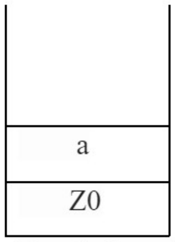
Initial



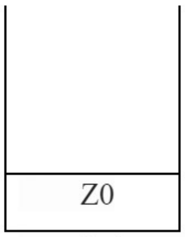
Reads a and push a



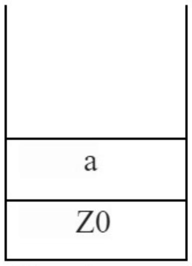
Reads a and push a



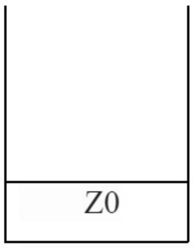
Reads b and pop out a



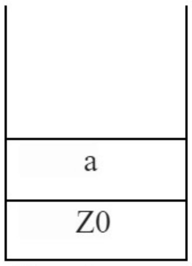
Reads b and pop out a



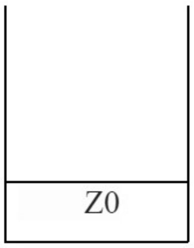
Reads a and push a



Reads b and pop out a

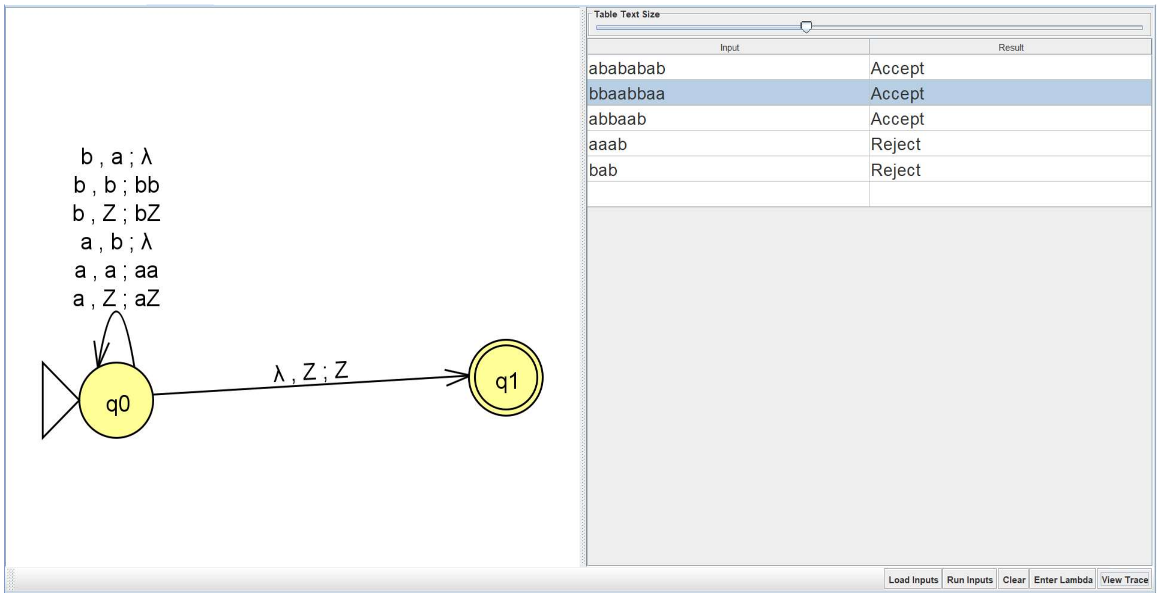


Reads a and push a

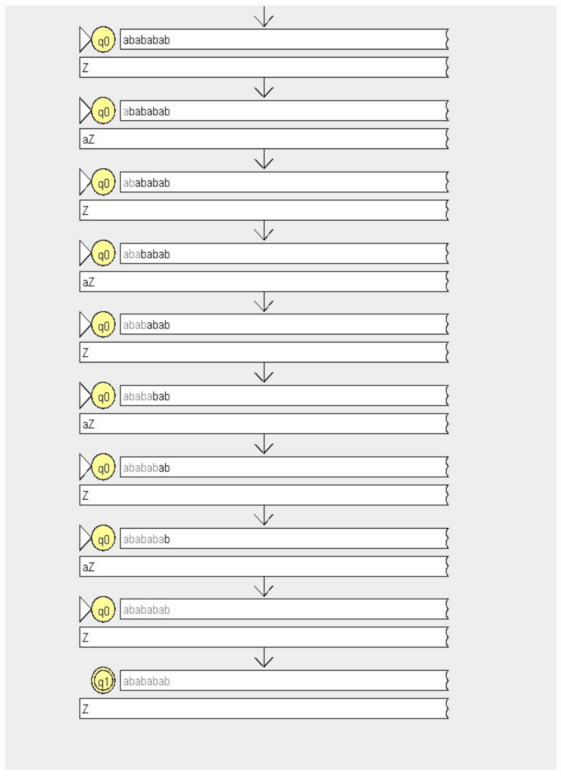


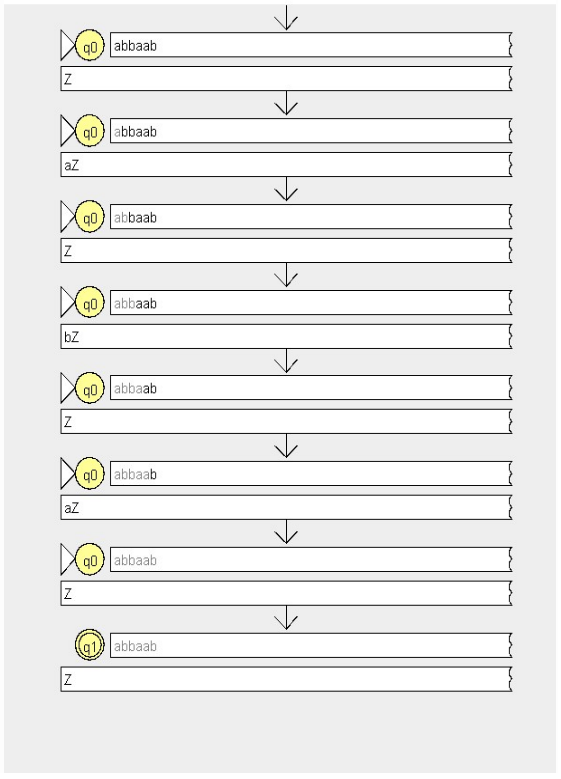
Reads b and pop out a

JFLAP Tracing for other input strings:



Input	Result
abababab	Accept
bbaabbaa	Accept
abbaab	Accept
aaab	Reject
bab	Reject

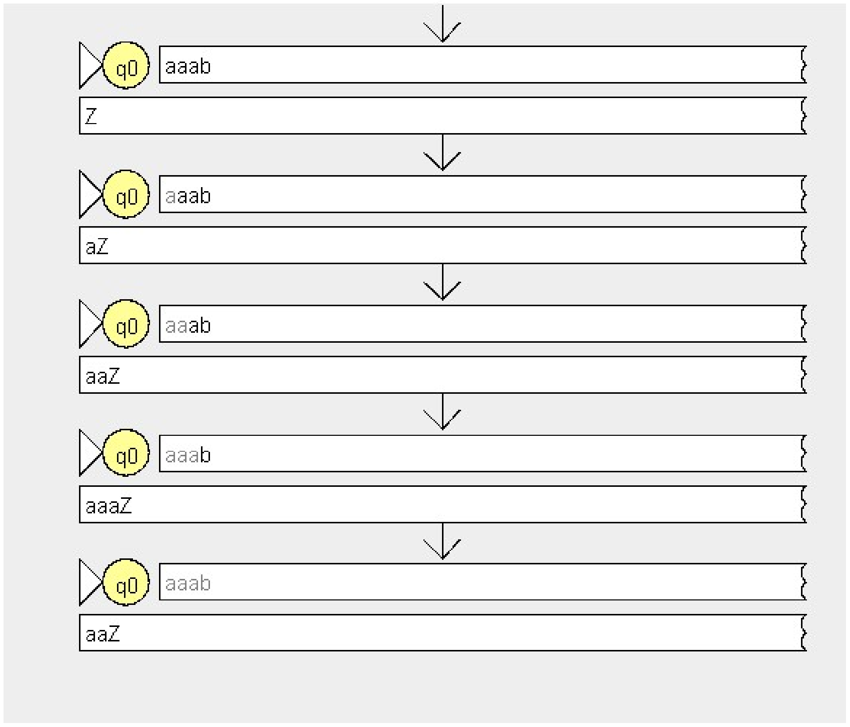




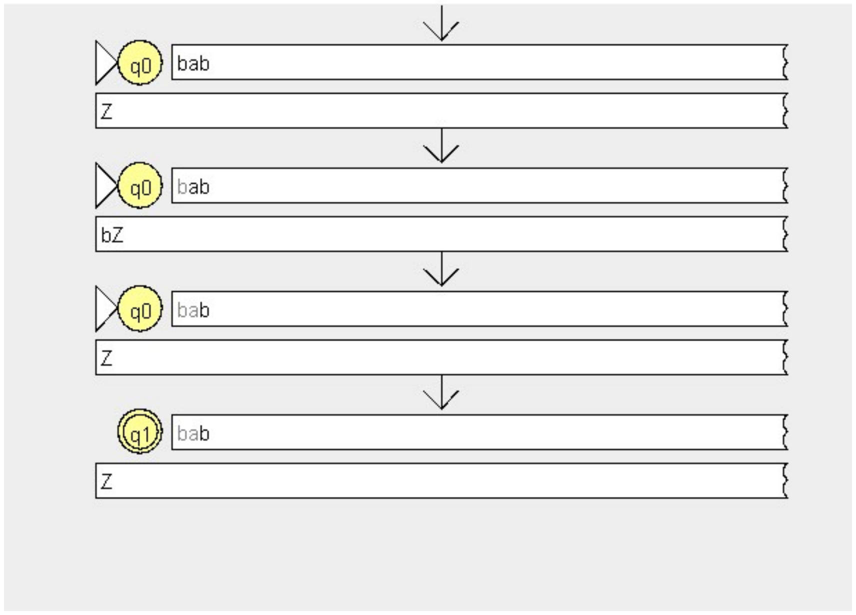
Input: abababab Input: abbaab

JFLAP Tracing for invalid strings

Input: aaab



Input: bab


q0
bab
Z
q0
bab
bZ
q0
bab
Z
q1
bab
Z