BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, V.V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Automata Theory and Computability Project (18CS54)

USN	Name	Batch	Phone	Email
1BI20CS033	Ashish Kumar Shukla	C1	9353092157	ashishkumar.12shukla@gmail.com

Table of Contents:

1. Deterministic Finite Automata
   L = {w Ɛ {a, b} : no two consecutive characters are the same}.*

2. Turing Machine
   Construct a Turing Machine that can accept the set of all even palindromes over {0,1}.

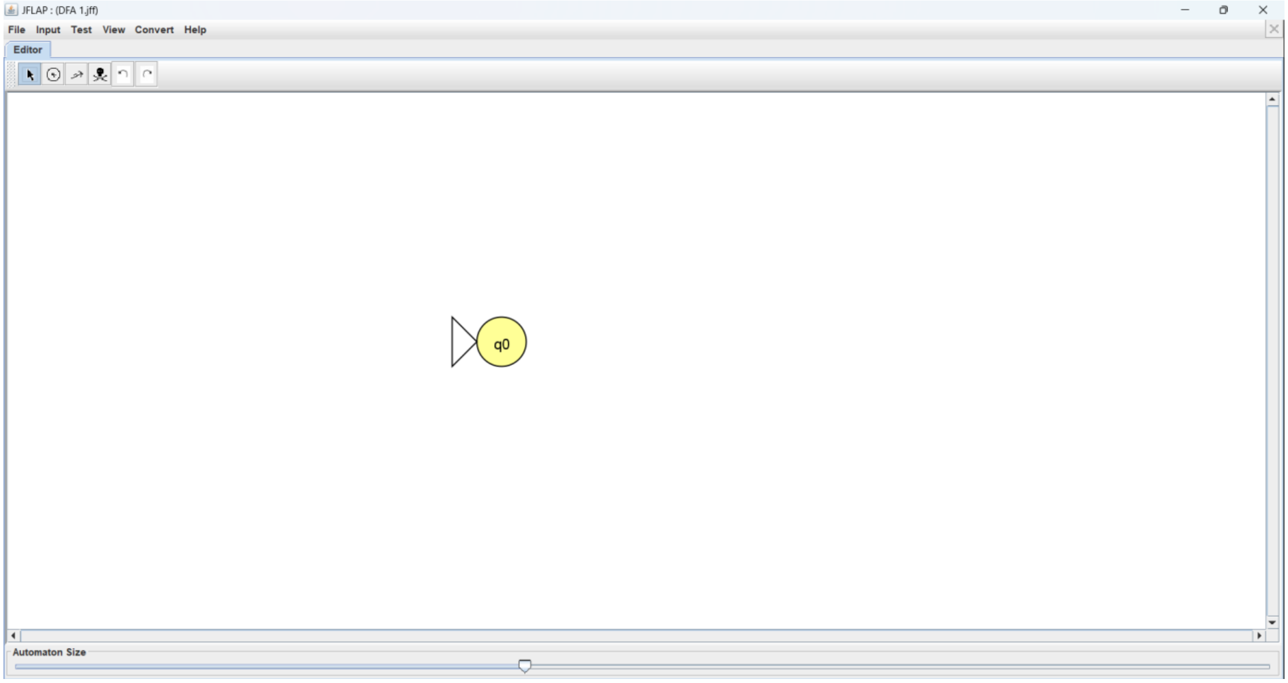
3. Grammer
   For the given Grammar G = S-> | aS | B , B-> | bB | [L = {w={a,b}, #a(w)=#b(w)}].*
   Construct a parse tree and derivation for the following string aaaabb.

4. Push Down Automata
   Construct a push down automata for the language L= { a^n b^n: n>=0}.

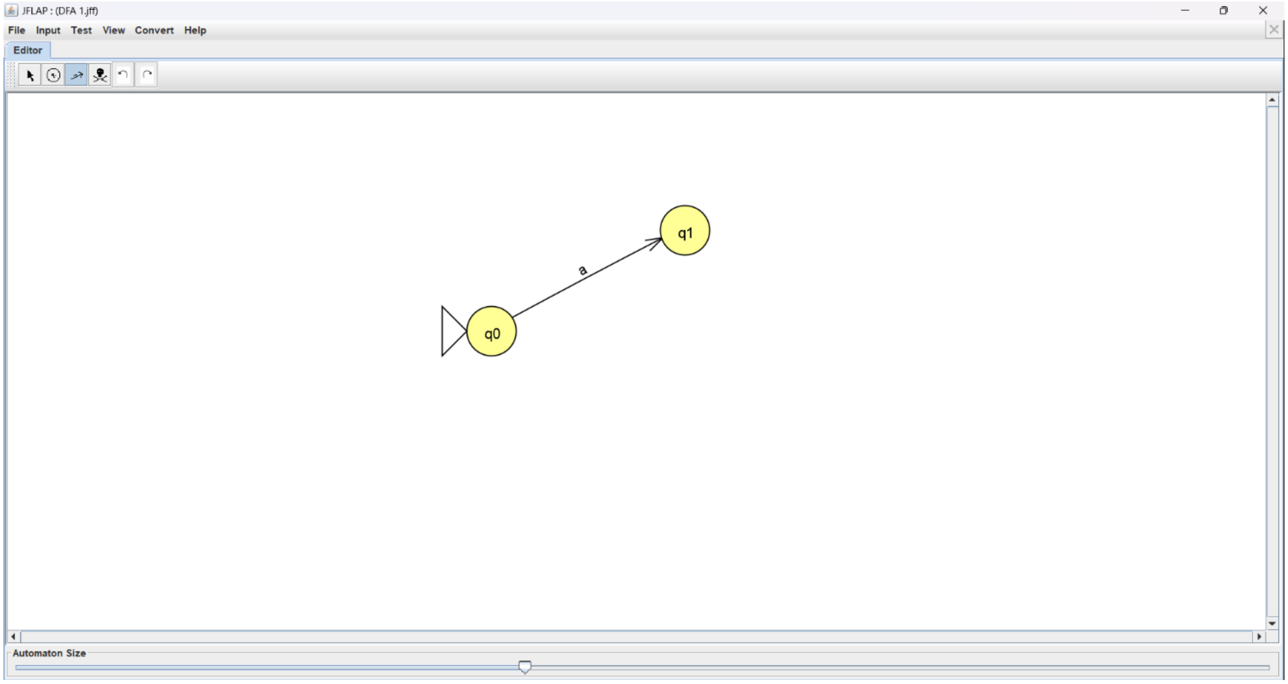
<u>DETERMINISTIC FINITE AUTOMATA:</u>

L = {w Ɛ {a. b}* : no two consecutive characters are the same}.

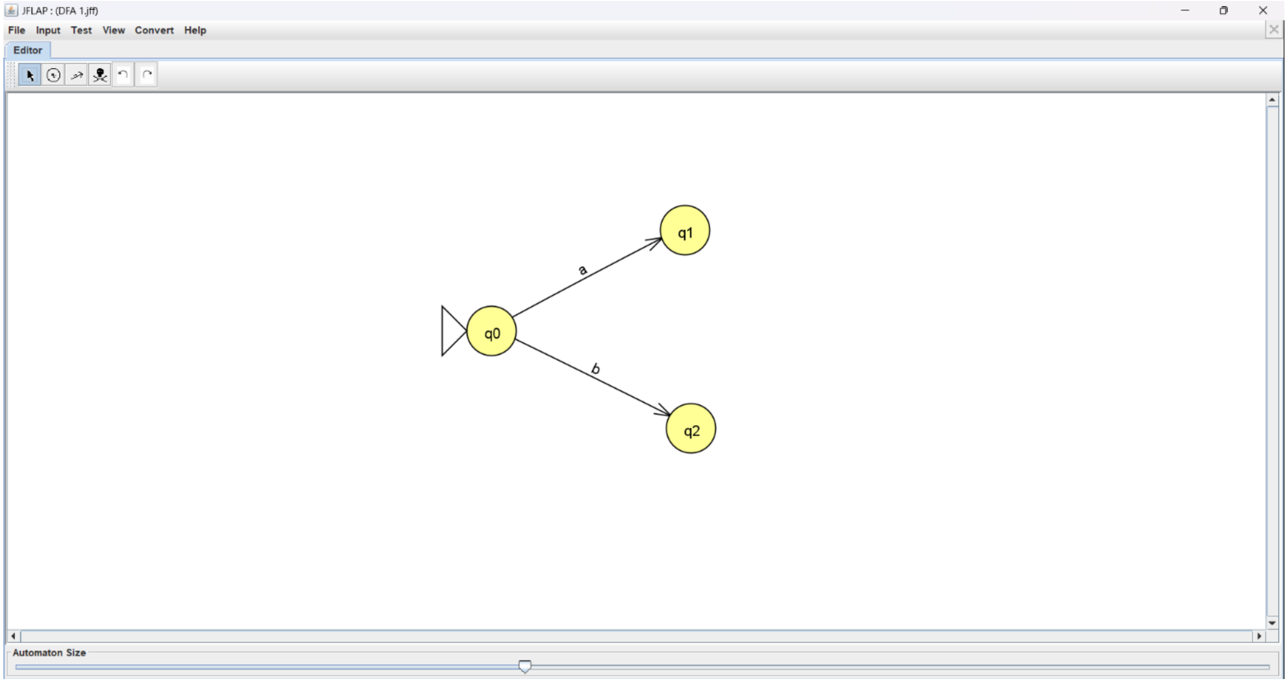
<u>Step 1</u>: q0 is the initial state of the machine.



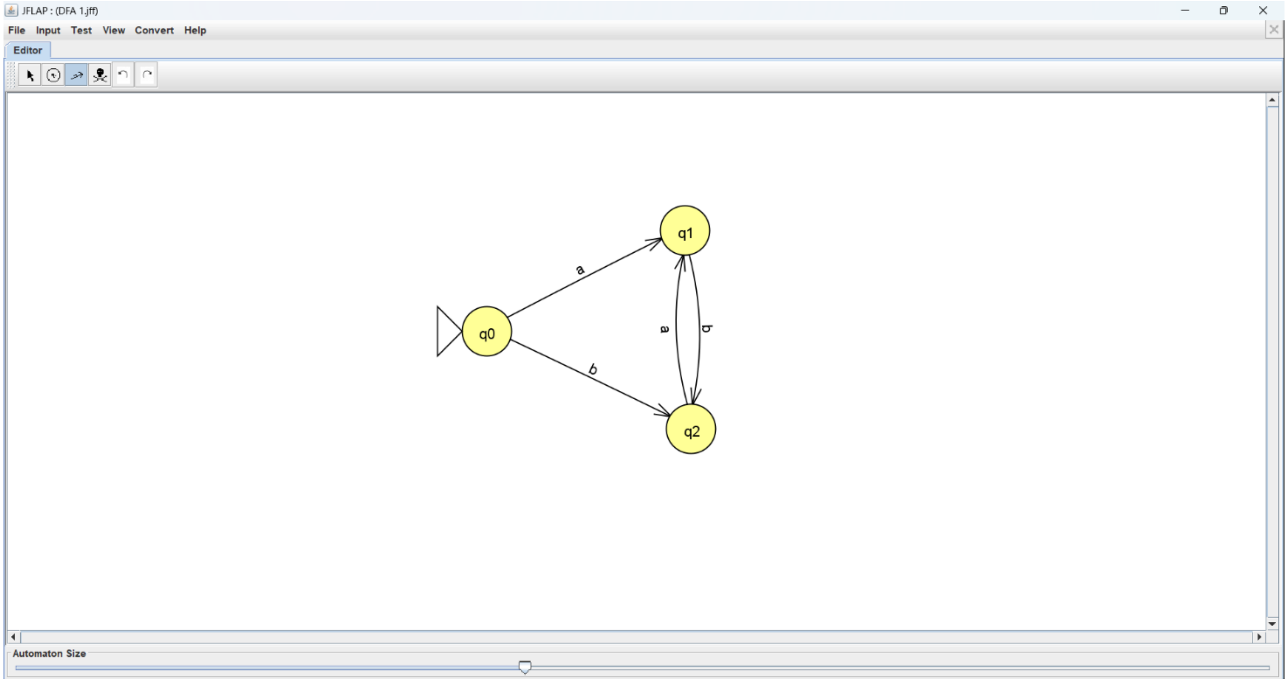
<u>Step 2</u>: Create state q1 to accept string ‘a’.



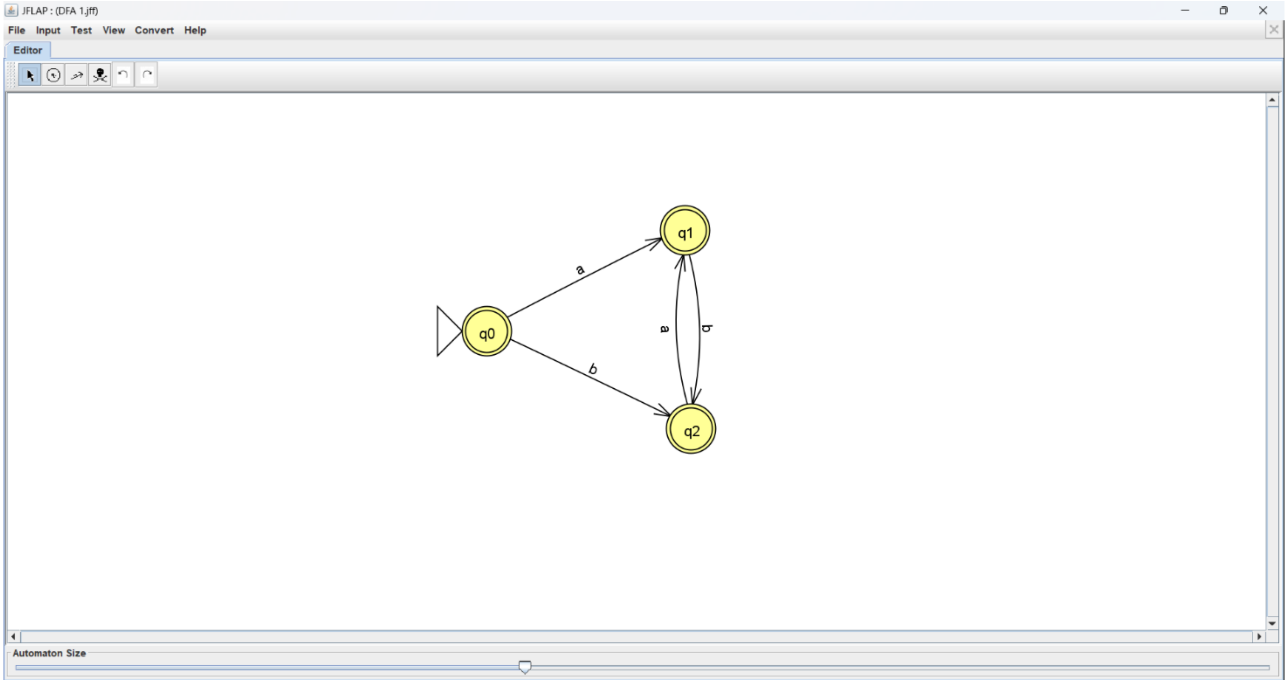
Step 3: Create state q2 to accept string ‘b’.



Step 4: State q1 goes to state q2 on accepting string b and state q2 goes to state q1 on accepting string a.

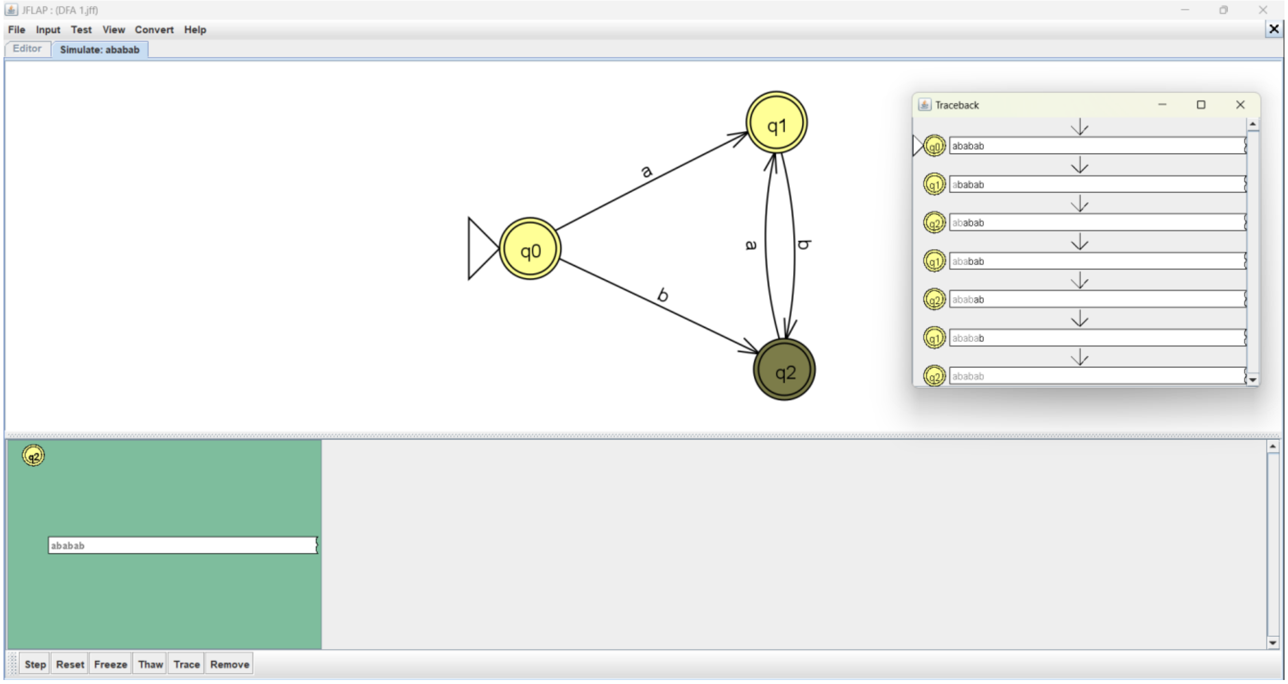


Step 5: State’s q0, q1 and q2 are the final state’s.

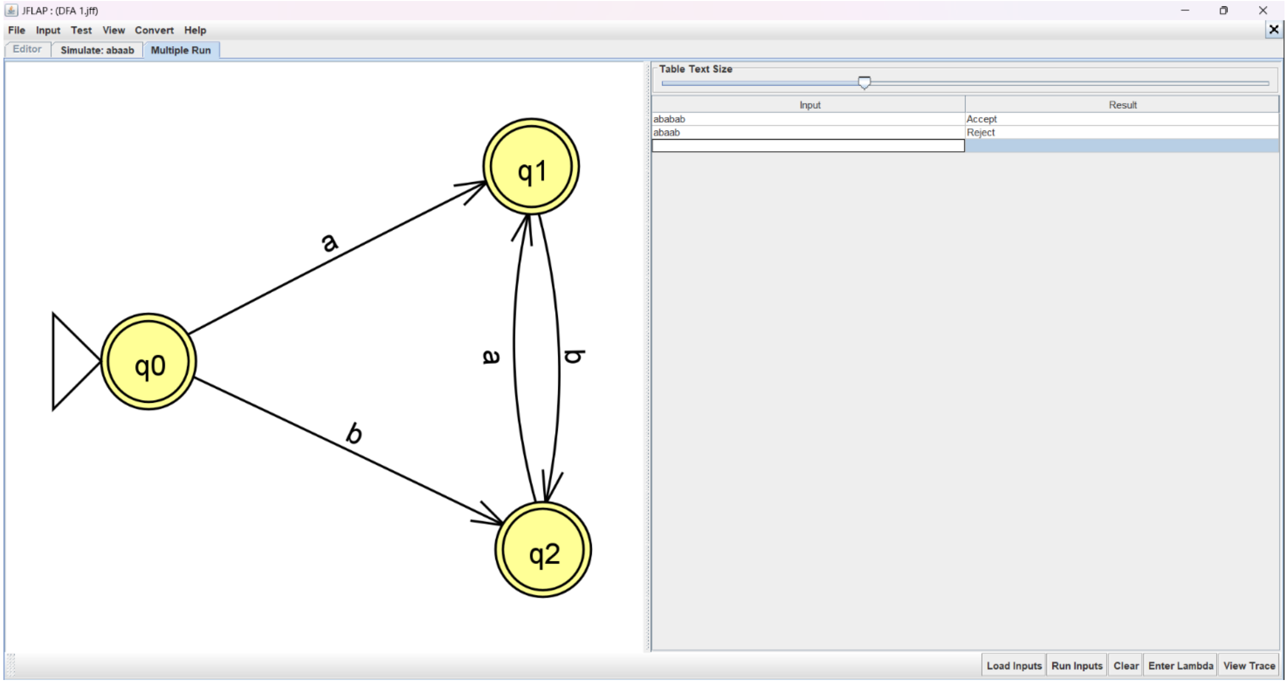


Trace

Tracing for string 'ababab' we get accepted as it satisfies the above DFSM condition.



When checking multiple strings, i.e., 'ababab' and 'abaab'. We observe that 'ababab' gets accepted and 'abaab' gets rejected. 'abaab' gets rejected because it doesn't follow the DFSM conditions above.

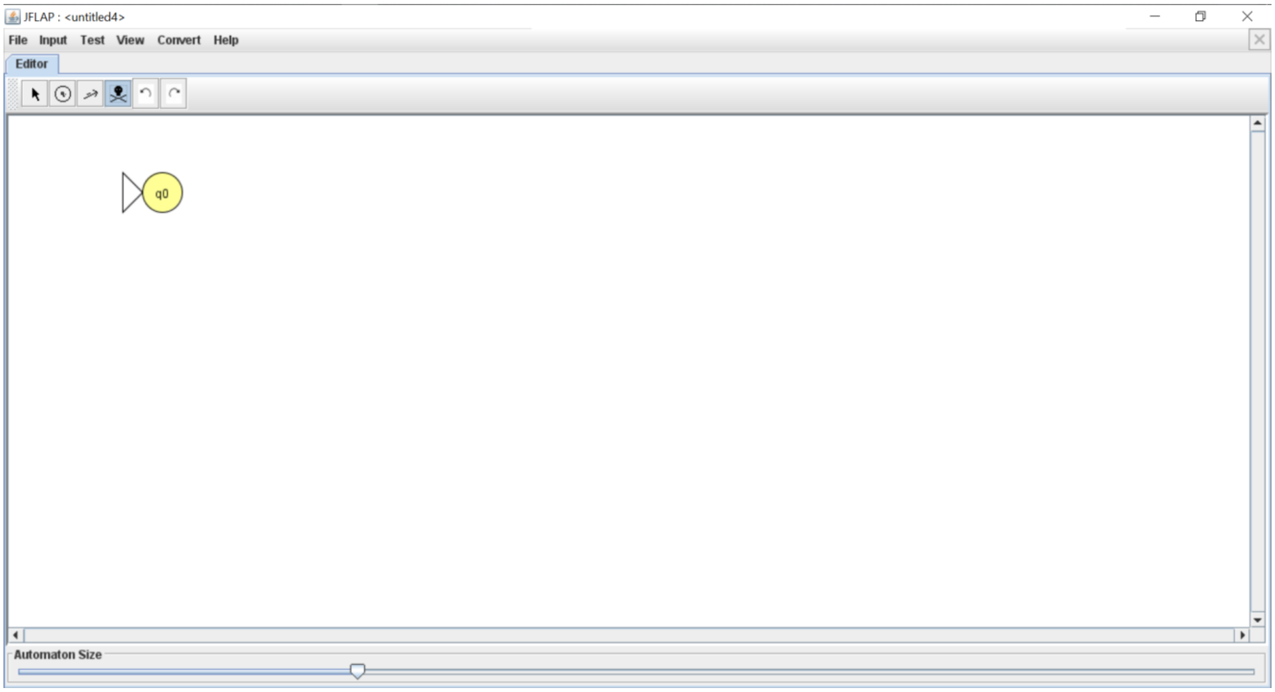


Input	Result
ababab	Accept
abaab	Reject

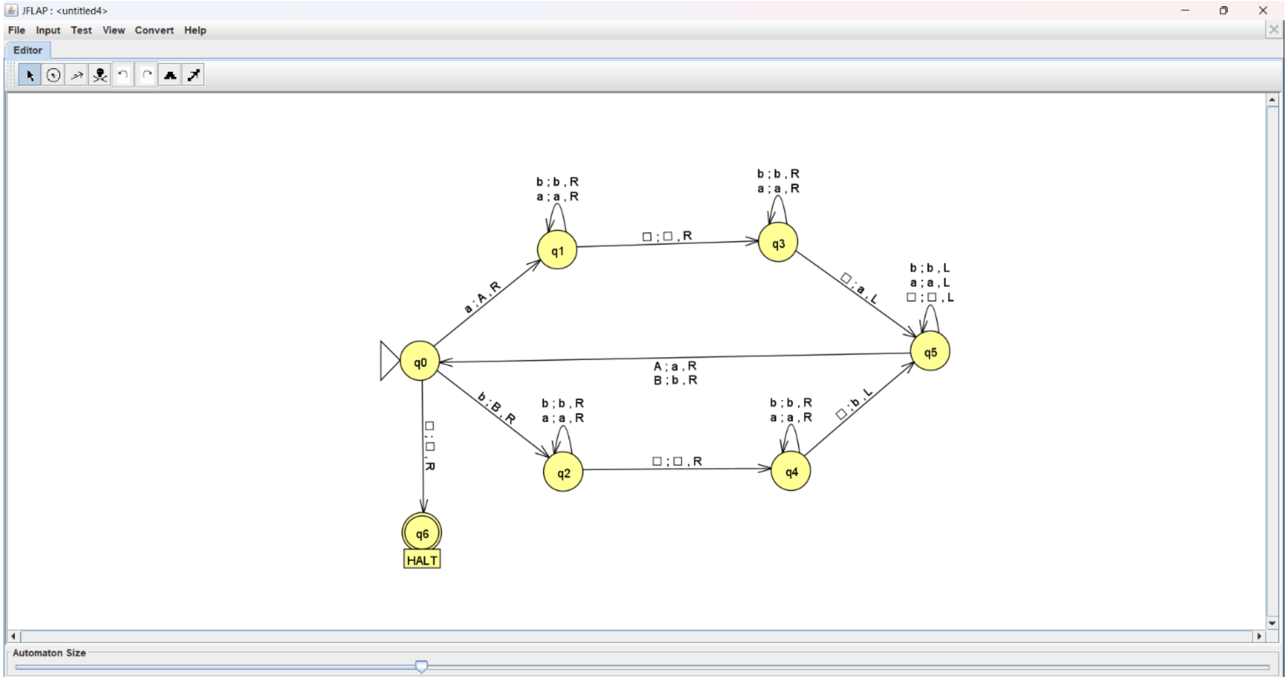
TURING MACHINE:

Construct a Turing Machine that can accept the set of all even palindromes over {0,1}.

Step 1: Create a new initial state q0. This is not the final state.



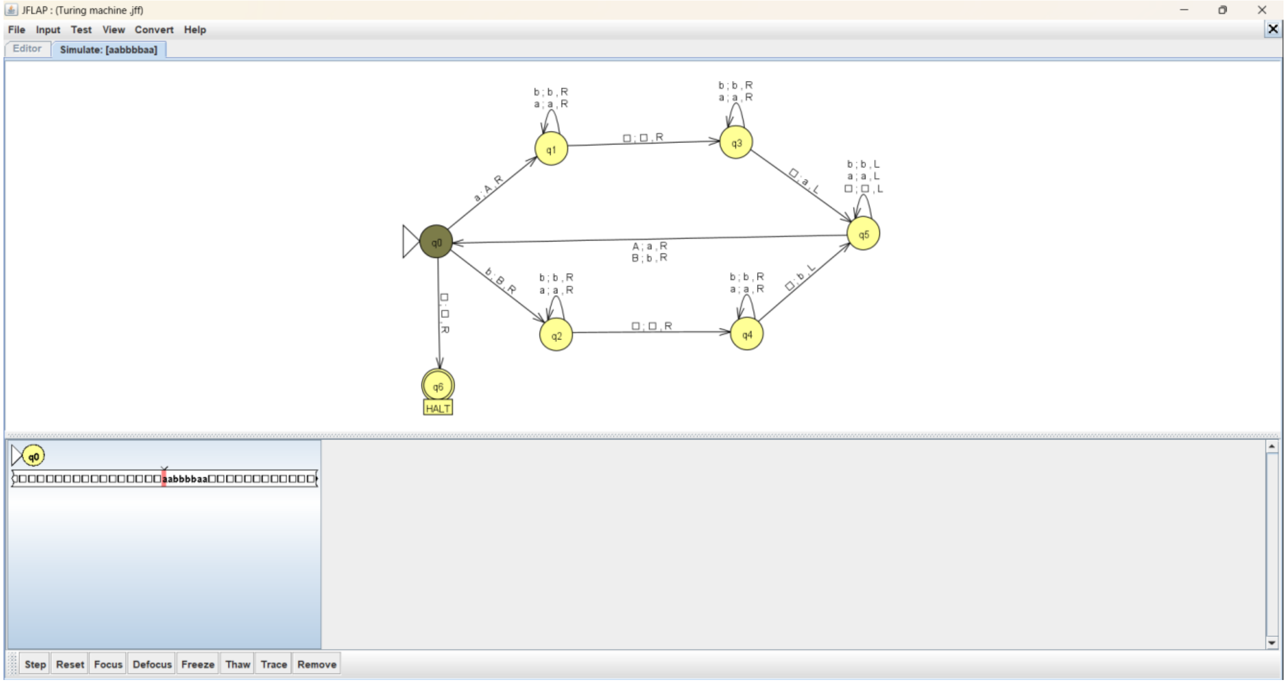
Step 2: This is the Transition Diagram for the required Turing Machine that can accept the set of all even palindromes over {0,1}..

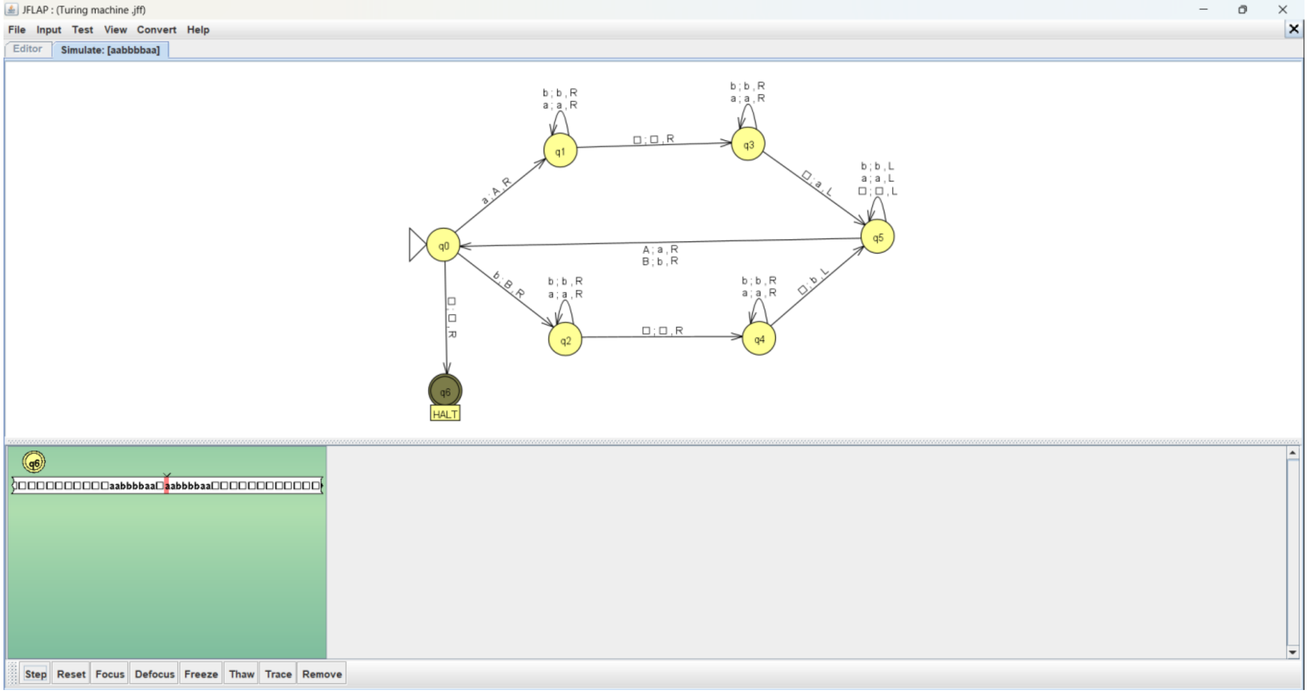


Ashish Kumar Shukla *1BI20CS033*

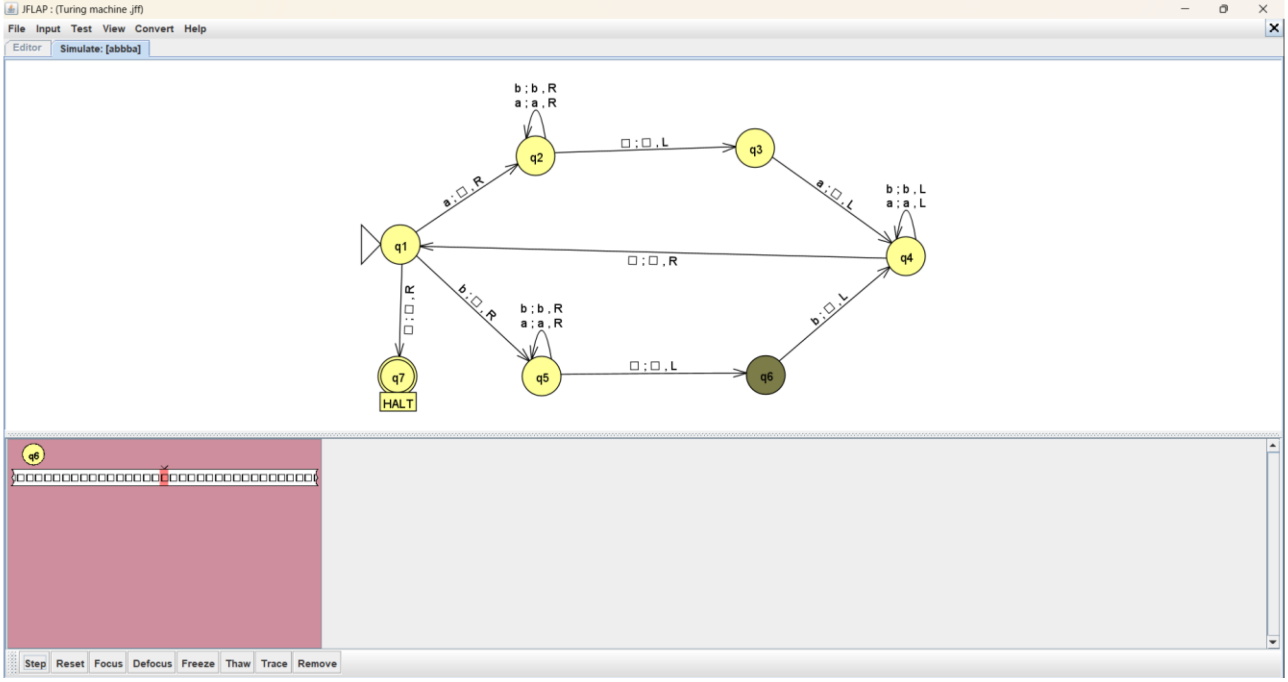
Trace

Tracing for ‘aabbbbaa’, we get an accepted state. Also, this a valid string according to our conditions.

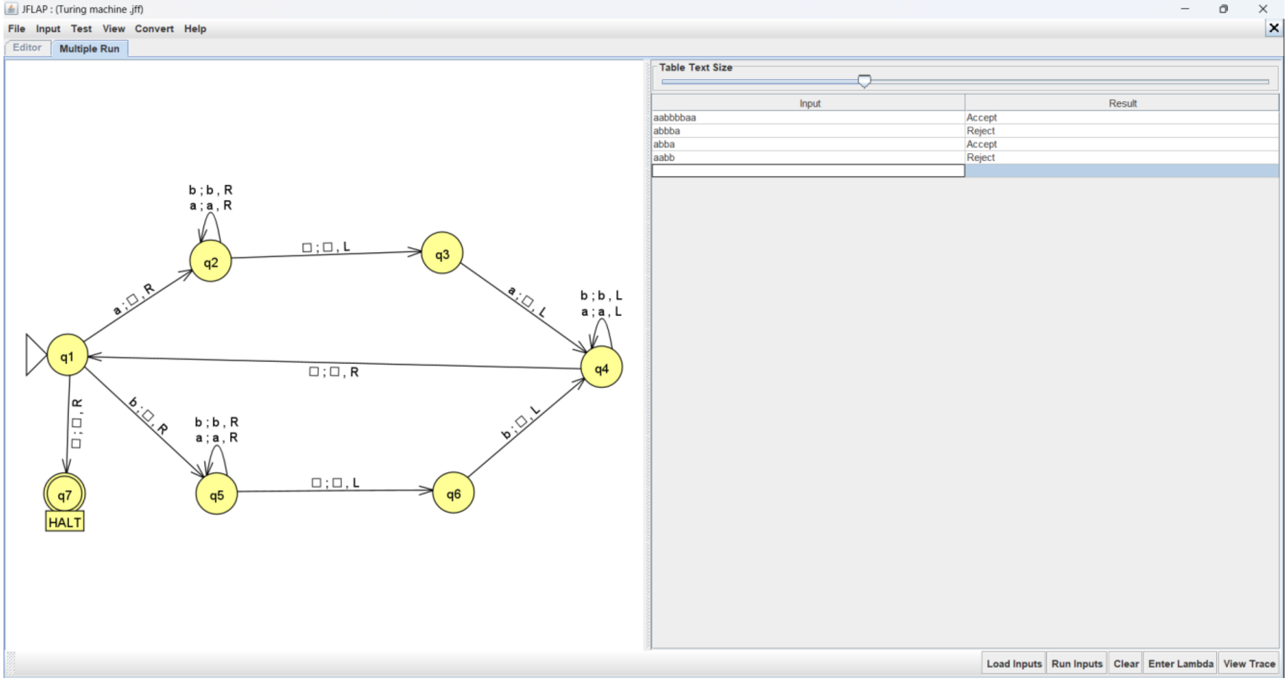




Input ‘abbba’ is rejected as it is an odd length palindrome.



For different type of inputs, we get these various answers -

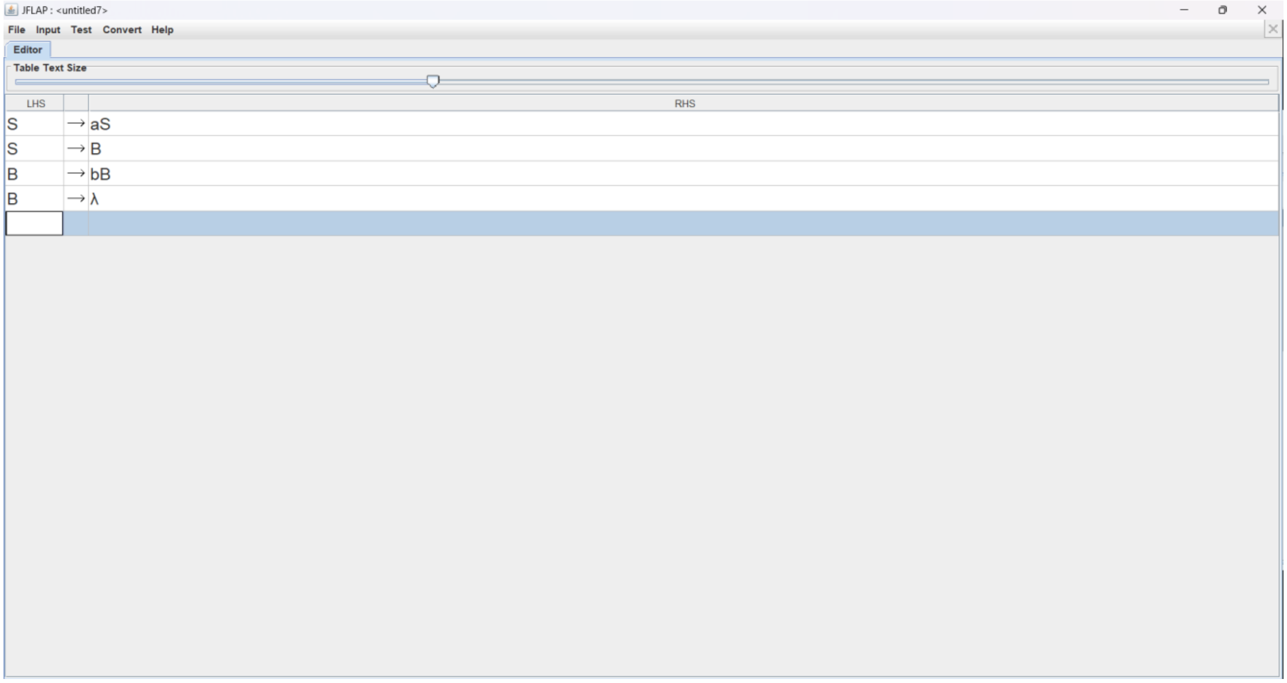


Input	Result
aabbbbaa	Accept
abbba	Reject
abba	Accept
aabb	Reject

GRAMMER

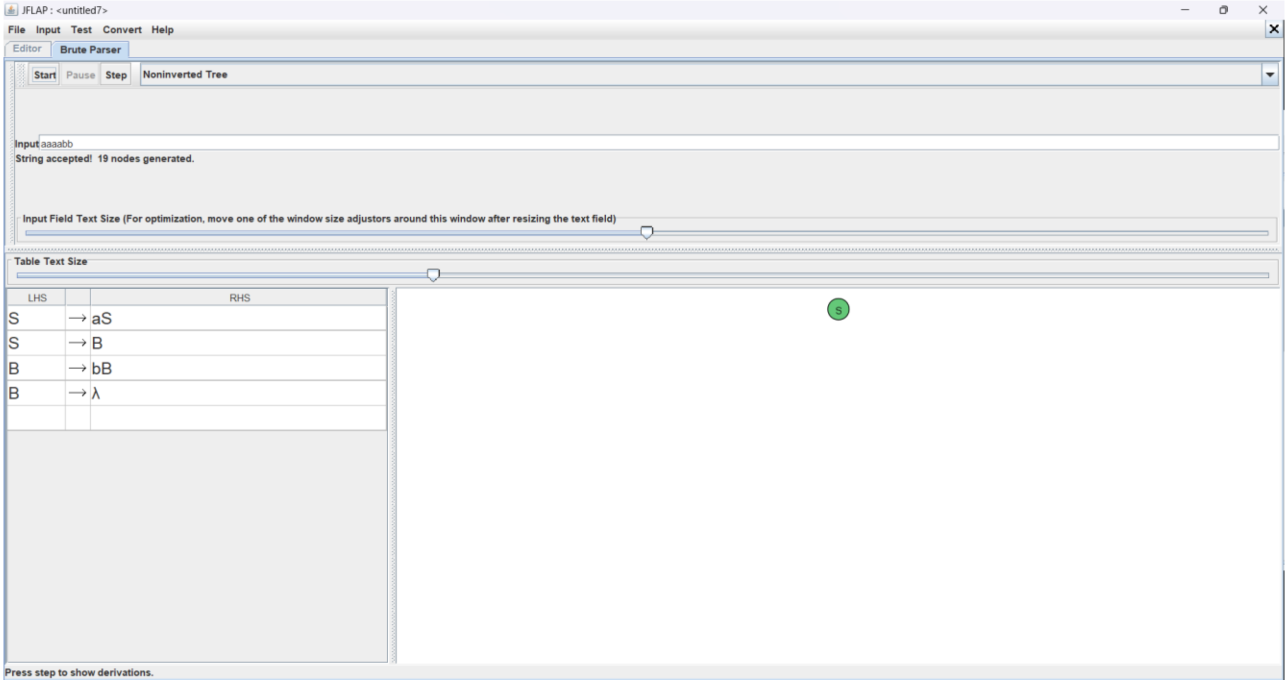
For the given Grammar G = S-> | aS | B , B-> | bB | [L = {w={a,b}*, #a(w)=#b(w)}]. Construct a parse tree and derivation for the following string aaaabb.

Step 1: The given Grammer is -



LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Step 2: S is the root node of the parse tree, which is a non-terminal.


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Brute Parser
Start
Pause
Step
Noninverted Tree
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
S

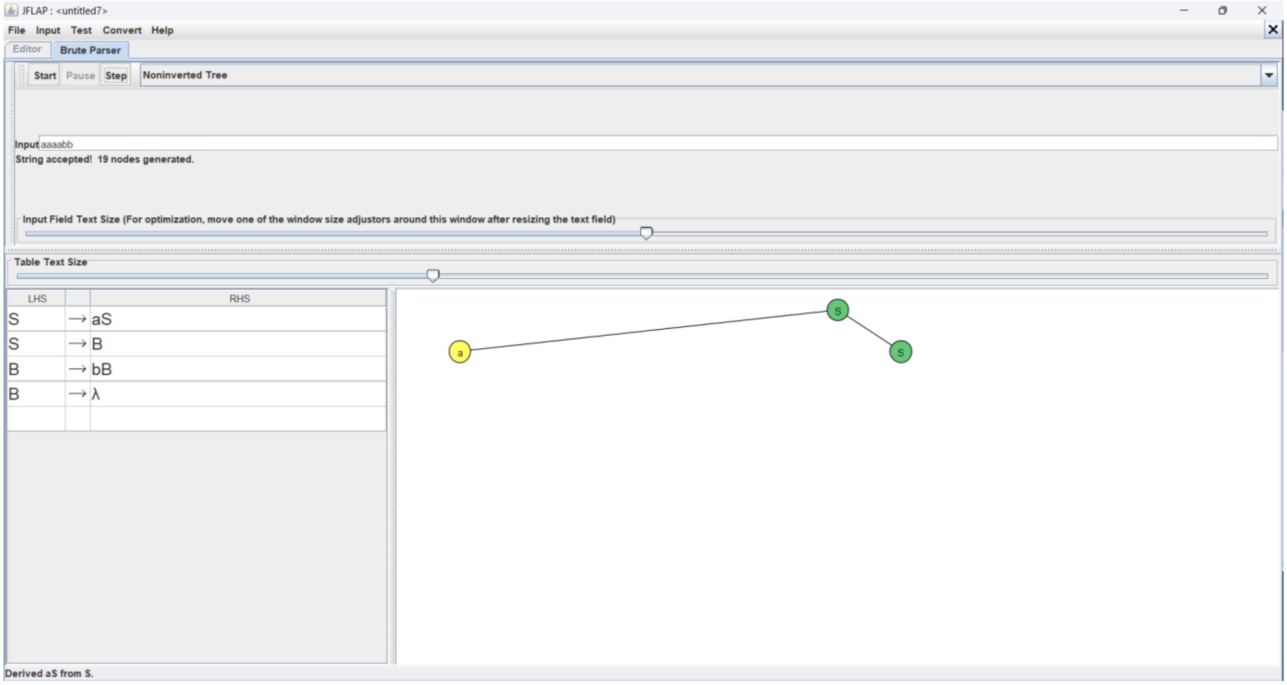
Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Press step to show derivations.

<u>Step 3</u>: The input string aaaabb is split into three sub trees. Hence the root node splits into a, S and S.


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Brute Parser
Start
Pause
Step
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
S
a
S

Noninverted Tree

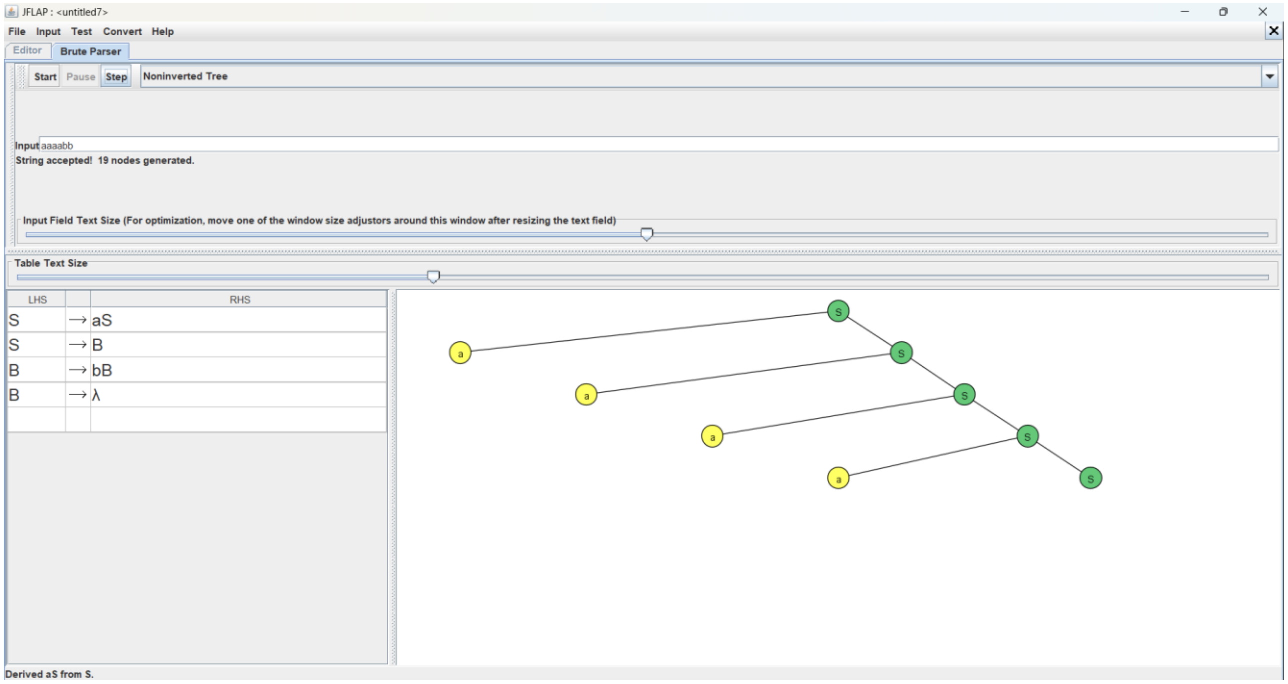
Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived aS from S.

<u>Step 4</u>: The mid tree S which is aaab further splits into aS and hence has three branches towards mid.


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Brute Parser
Start
Pause
Step
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
S
a
S
a
S
a
S
a
S

Noninverted Tree

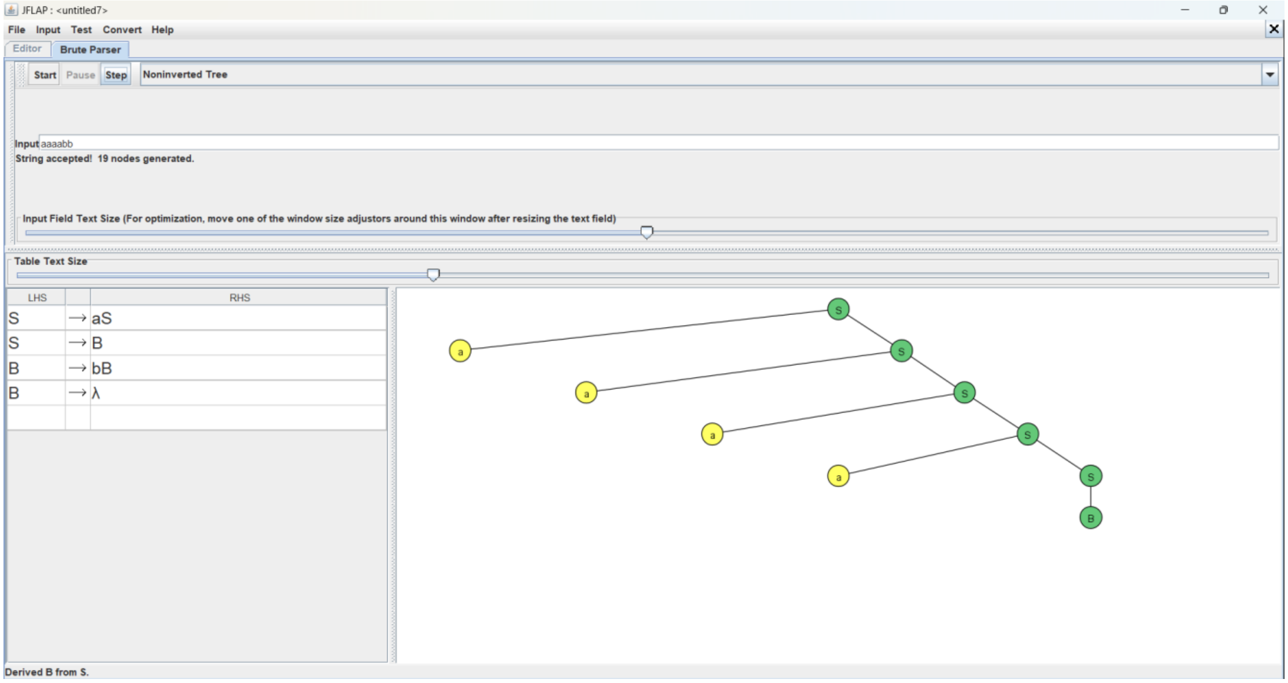
Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived aS from S.

Step 5: The S in 4th level which has aab splits into SB and hence has one branches B.


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Brute Parser
Start Pause Step
Noninverted Tree
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
S
a
S
a
S
a
S
a
S
B

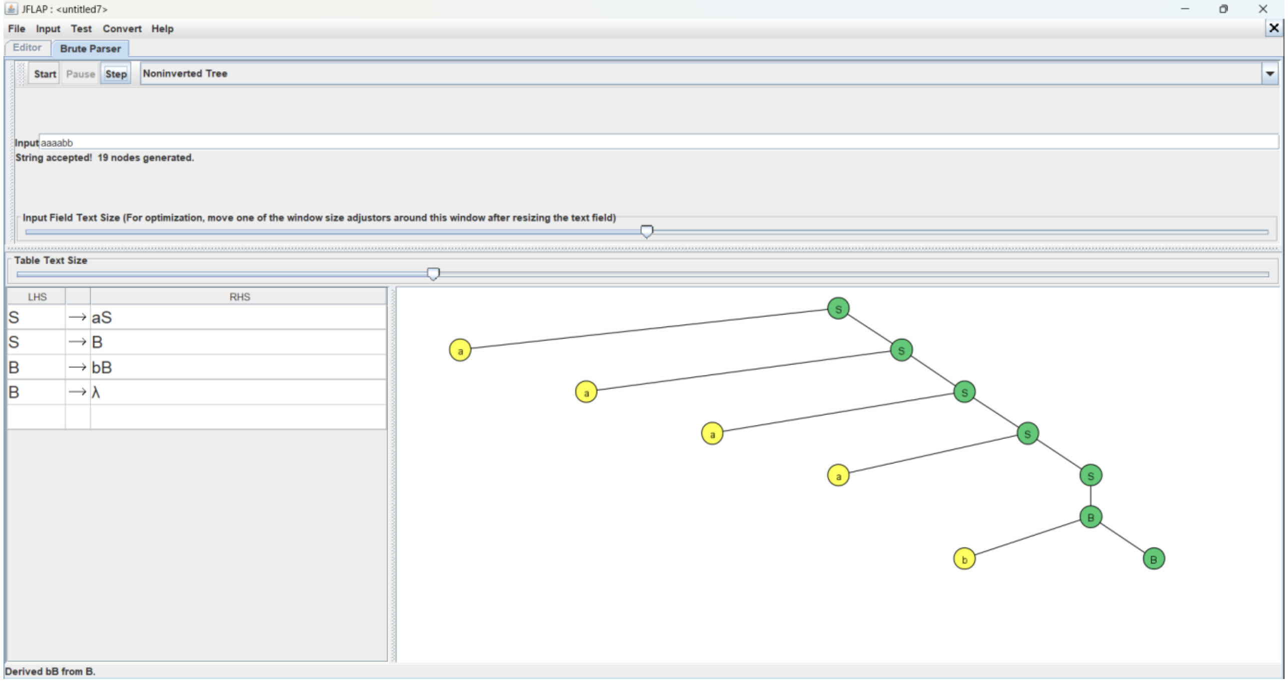
Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived B from S.

Step 6: The B in 5th level further splits into bB and hence has two branches b and B.


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Brute Parser
Start Pause Step
Noninverted Tree
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
S
a
S
a
S
a
S
a
S
B
b
B

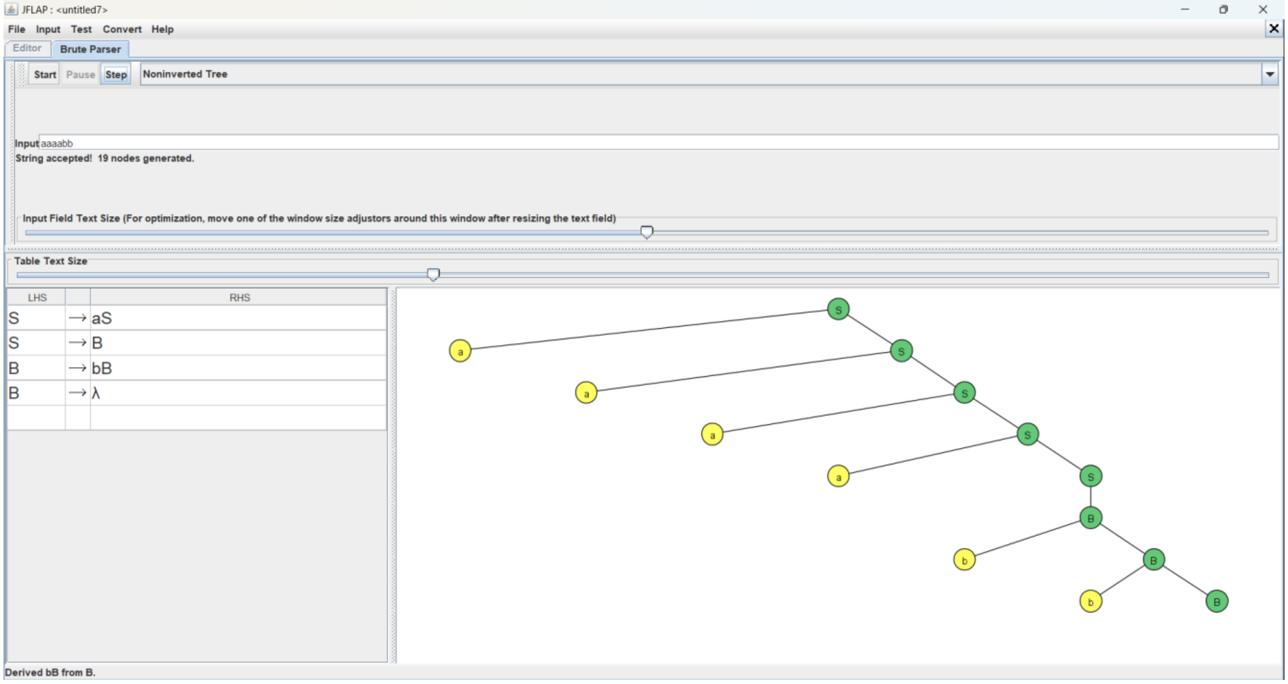
Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived bB from B.

Step 7: The B in the 7th level further splits into bB and hence has two branches b and B.



Input aaaabb

String accepted! 19 nodes generated.

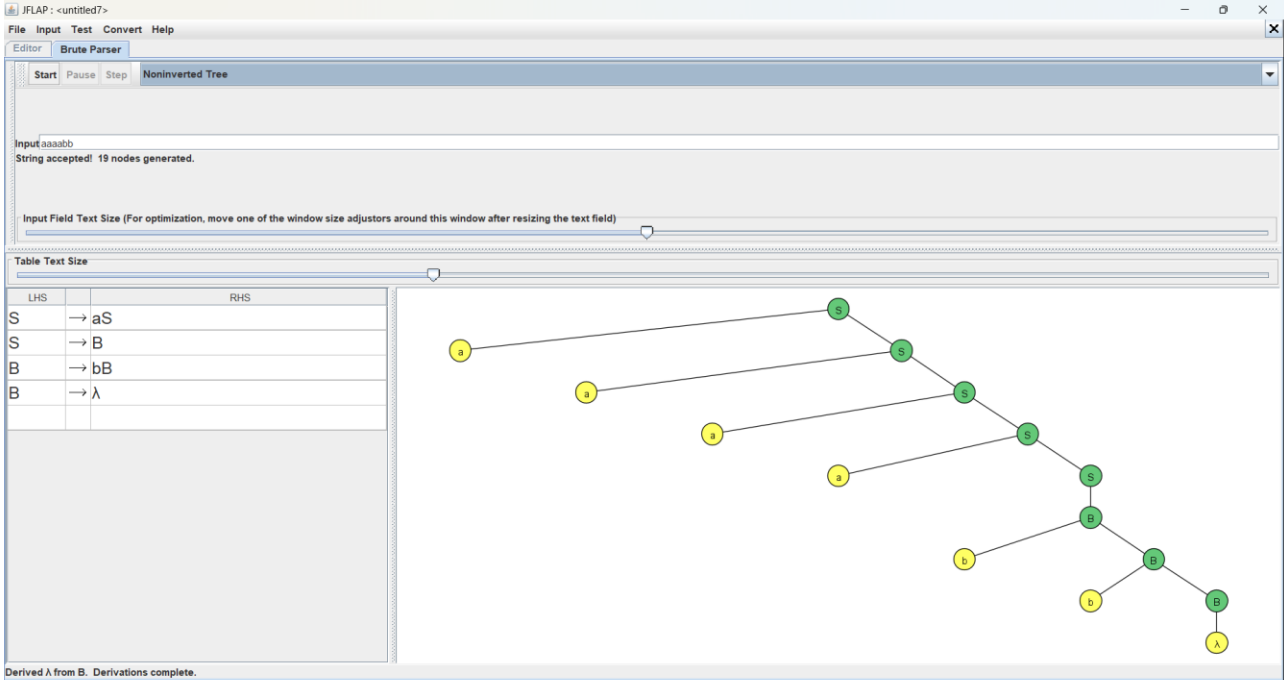
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)

Table Text Size

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived bB from B.

Step 8: The B can now have an empty string.



Input aaaabb

String accepted! 19 nodes generated.

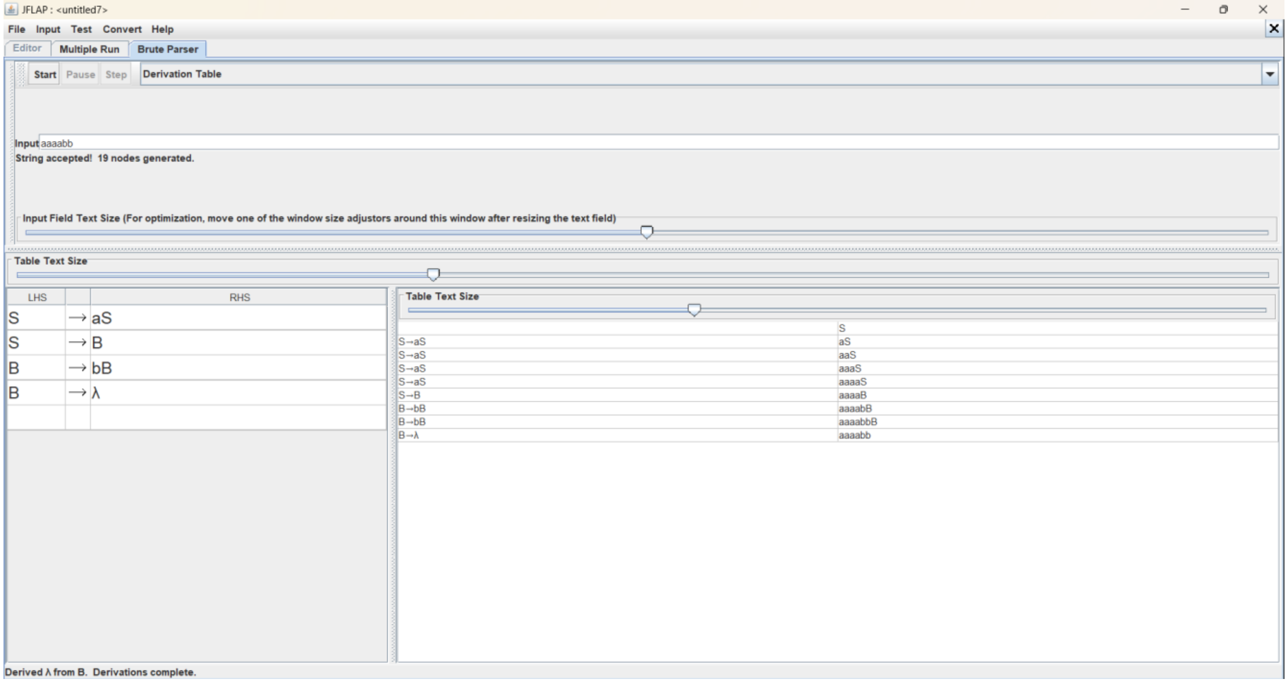
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)

Table Text Size

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

Derived λ from B. Derivations complete.

Derivation Table


JFLAP : <untitled7>
File Input Test Convert Help
Editor
Multiple Run
Brute Parser
Start
Pause
Step
Derivation Table
Input Field Text Size (For optimization, move one of the window size adjustors around this window after resizing the text field)
Table Text Size
Table Text Size

Input aaaabb

String accepted! 19 nodes generated.

LHS		RHS
S	→	aS
S	→	B
B	→	bB
B	→	λ

	S
S→aS	aS
S→aS	aaS
S→aS	aaaS
S→aS	aaaaS
S→B	aaaaB
B→bB	aaaabB
B→bB	aaaabbB
B→λ	aaaabb

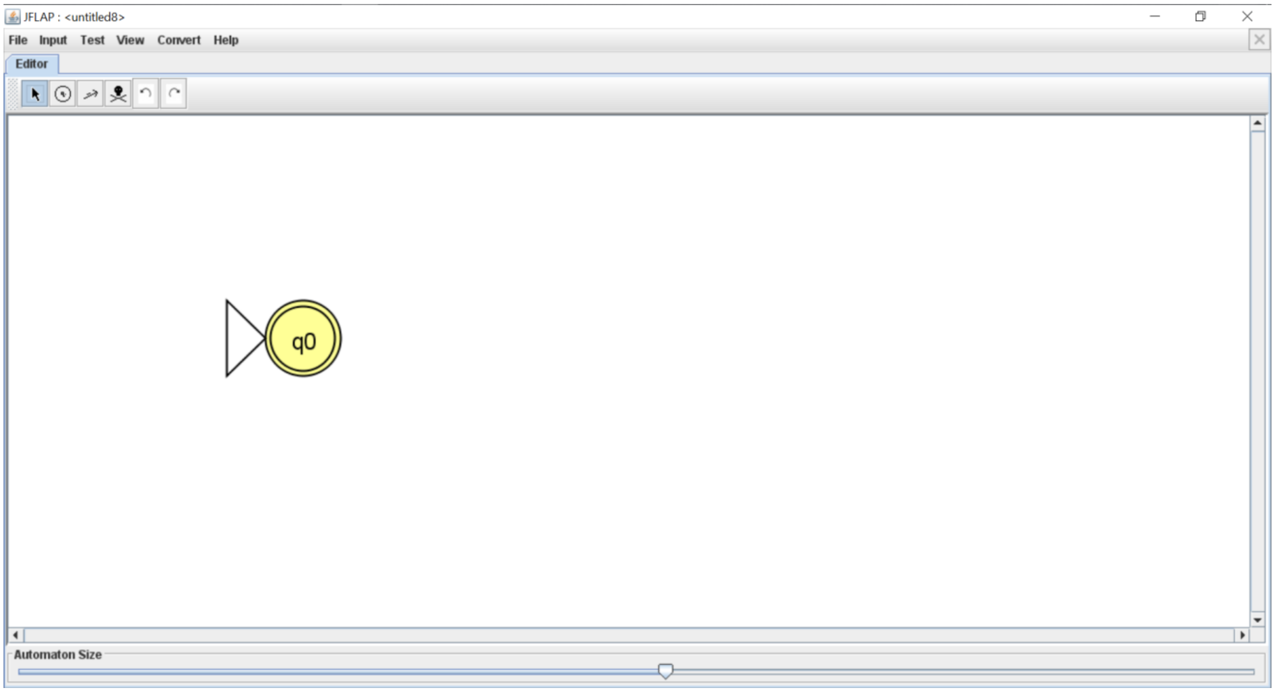
Derived λ from B. Derivations complete.

Since the string could be derived using a parse tree and derivation table, the string aaaabb is accepted.

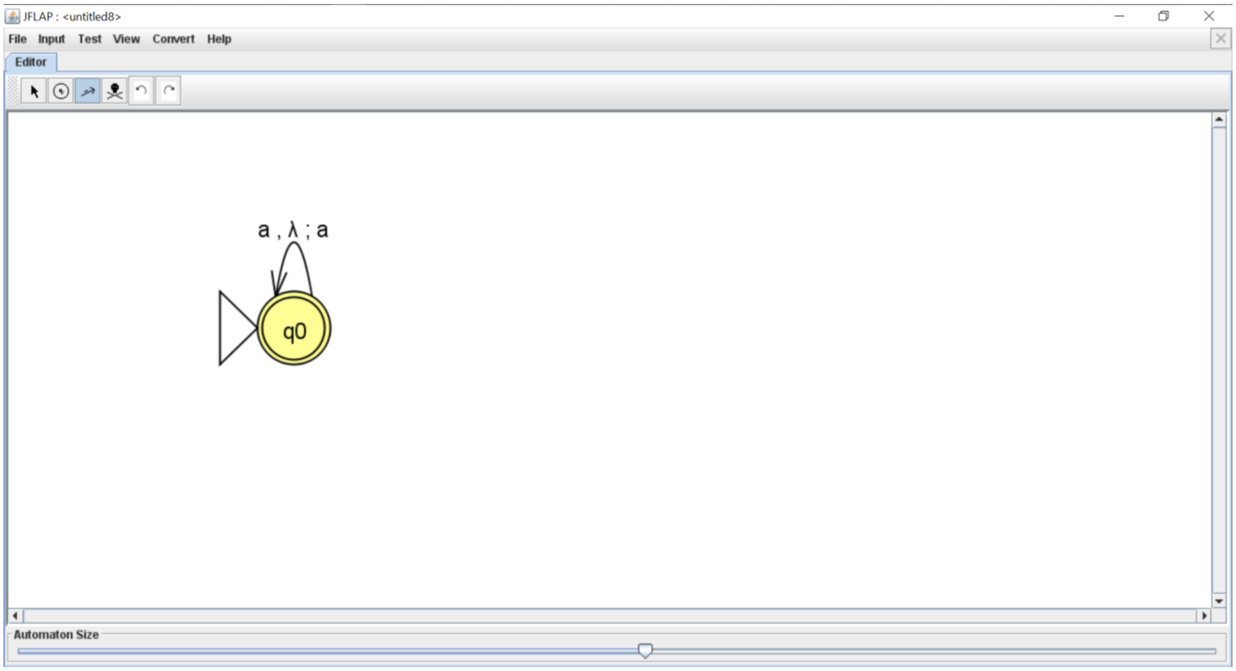
PUSH DOWN AUTOMATA

Construct a push down automata for the language L= { a^n b^n: n>=0}.

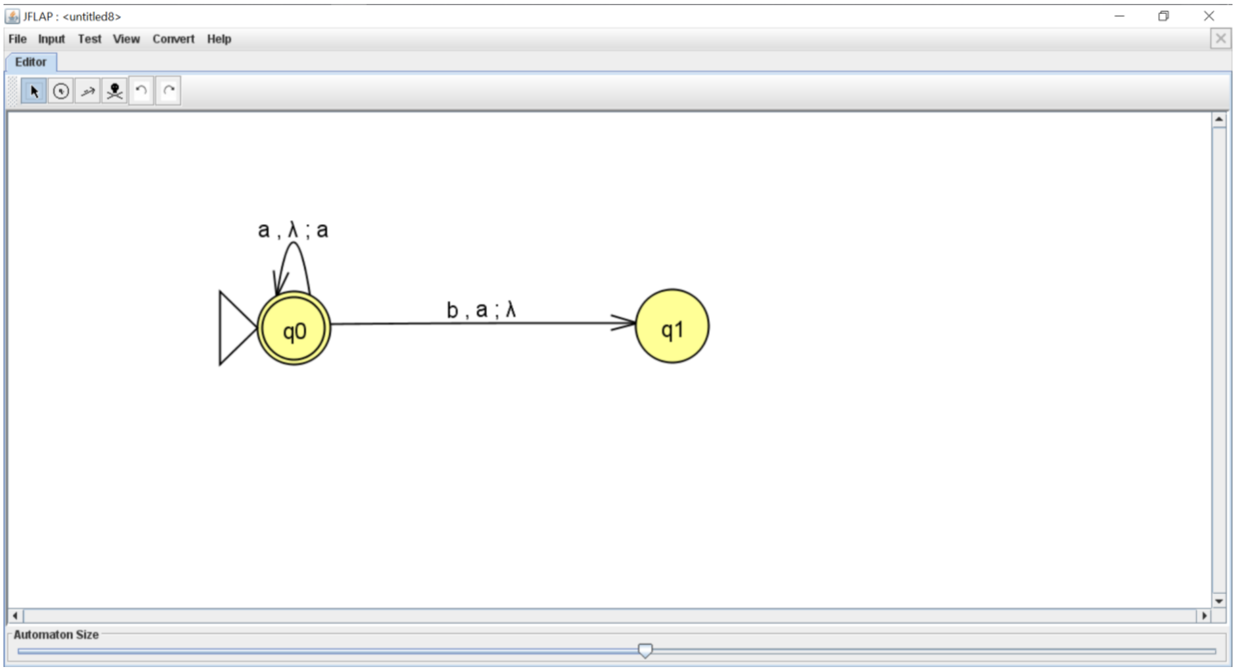
Step 1: Create a new initial state q0. It is a final state since the language doesn't accept empty strings.



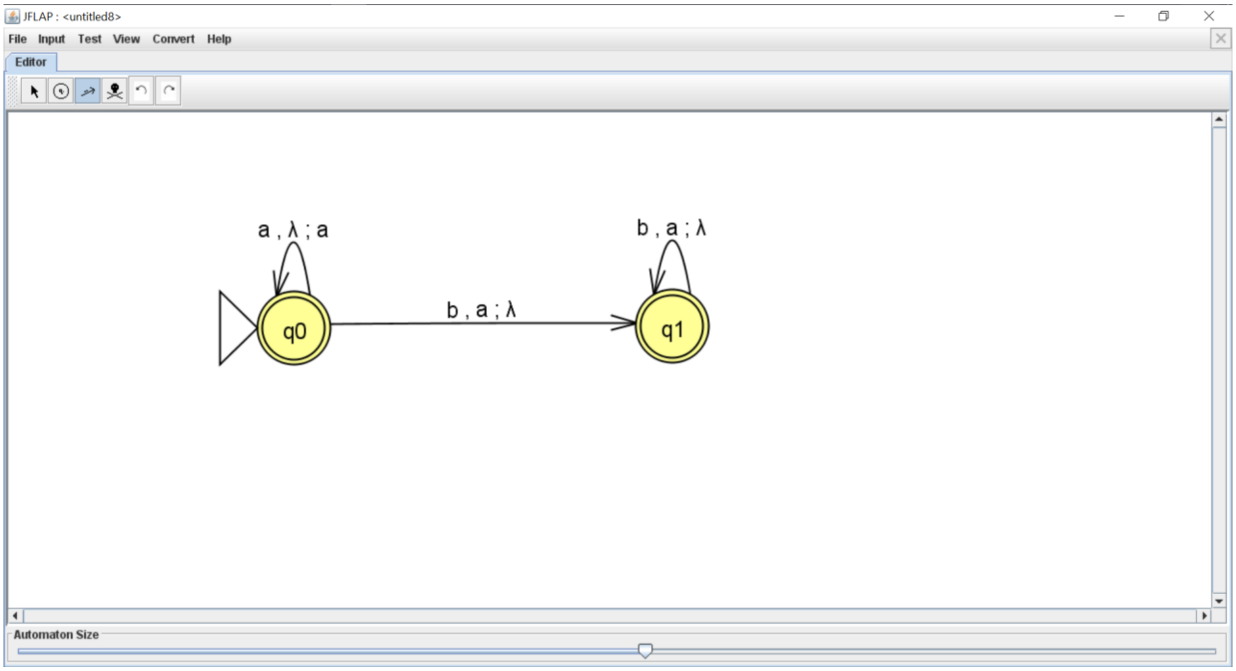
Step 2: Self-transition on q0 – if the input symbol is a and the stack is empty, a is pushed onto the stack.



Step 3: b transition from state q0 to q1.



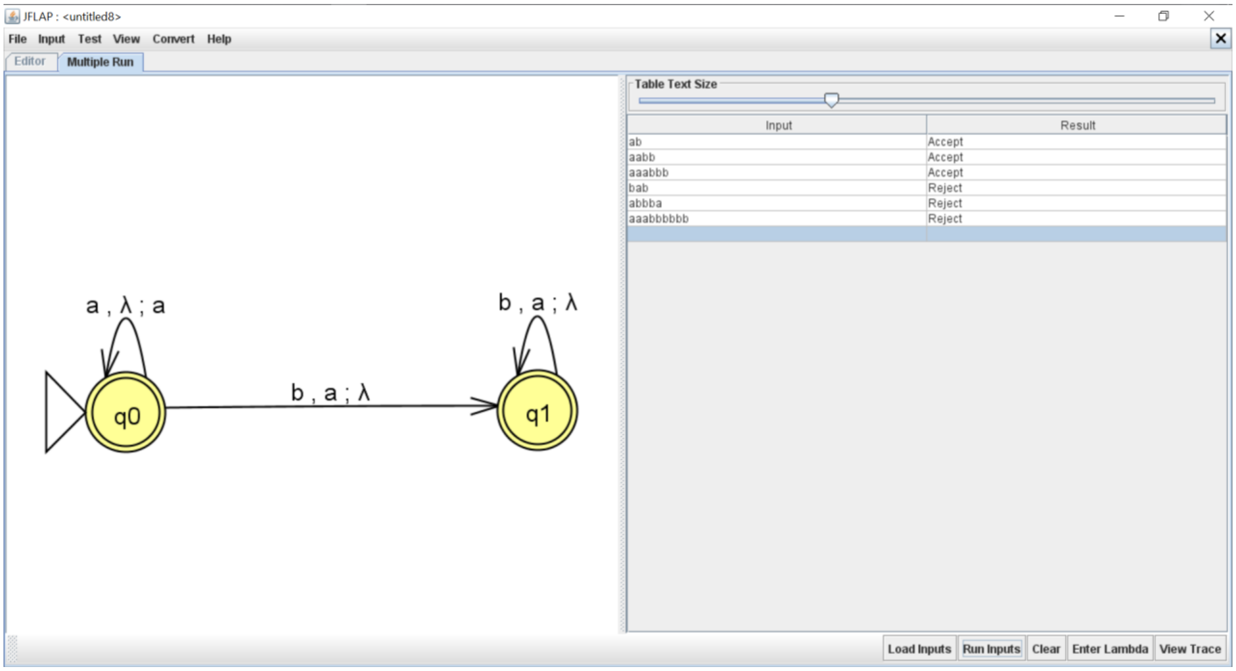
Step 4: State q1 is made the final state. Self-transition on q1 – if the input symbol is b and a is on top of the stack then symbol on top is popped off.



Step 6: Testing the PDA with multiple inputs and verifying the results.

Accepting strings: ab, aabb, aaabbb

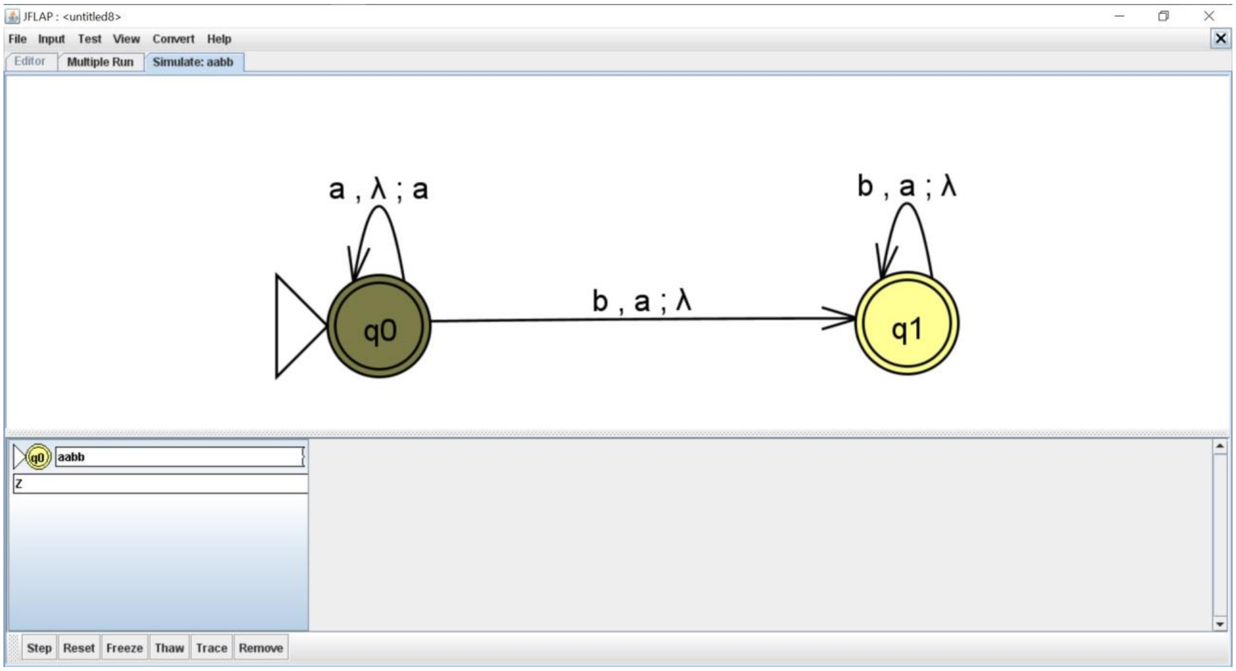
Rejecting strings: bab, abbba, baaab, aaabbbbbb

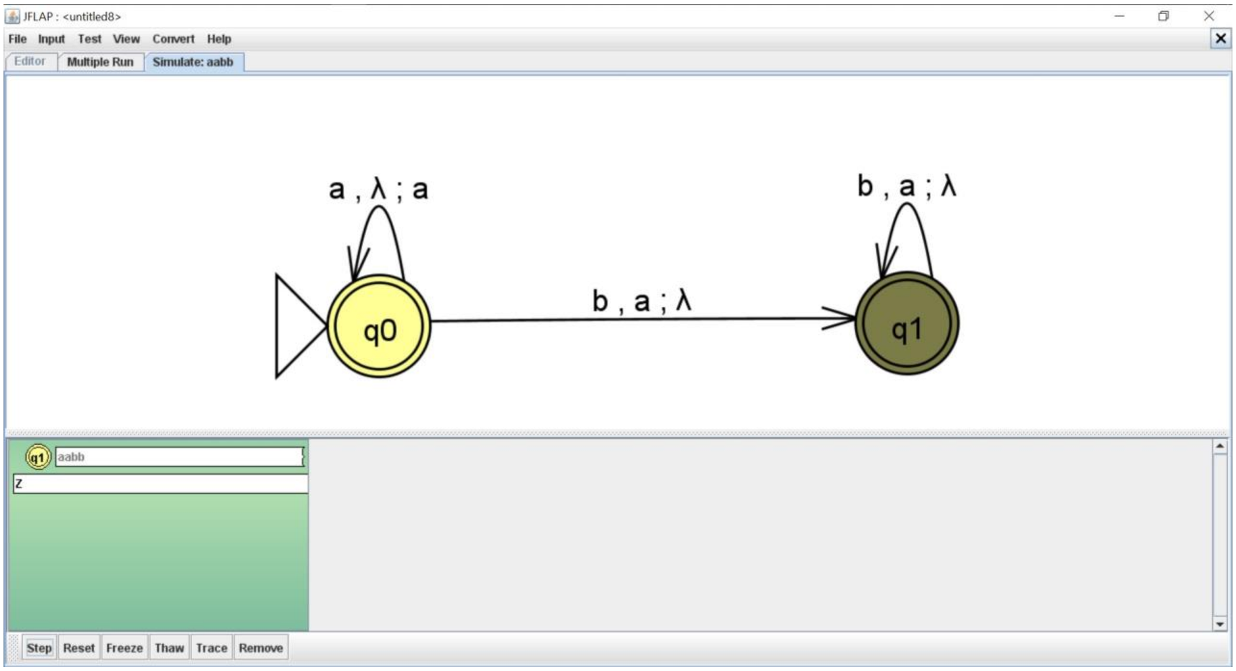


Input	Result
ab	Accept
aabb	Accept
aaabbb	Accept
bab	Reject
abbba	Reject
aaabbbbbb	Reject

Trace

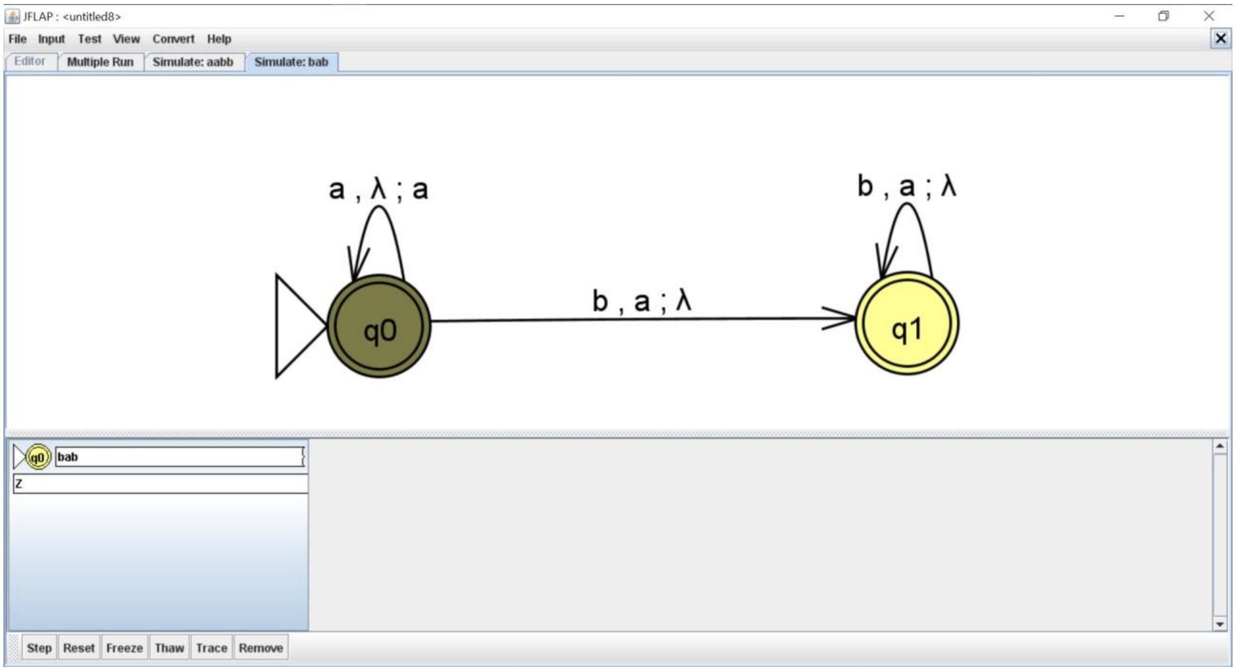
Accepting String: aabb


JFLAP : <untitled8>
File Input Test View Convert Help
Editor Multiple Run Simulate: aabb
a , λ ; a
b , a ; λ
b , a ; λ
q0
q1
aabb
Z
Step Reset Freeze Thaw Trace Remove


JFLAP : <untitled8>
File Input Test View Convert Help
Editor Multiple Run Simulate: aabb
a , λ ; a
b , a ; λ
b , a ; λ
q0
q1
aabb
Z
Step Reset Freeze Thaw Trace Remove

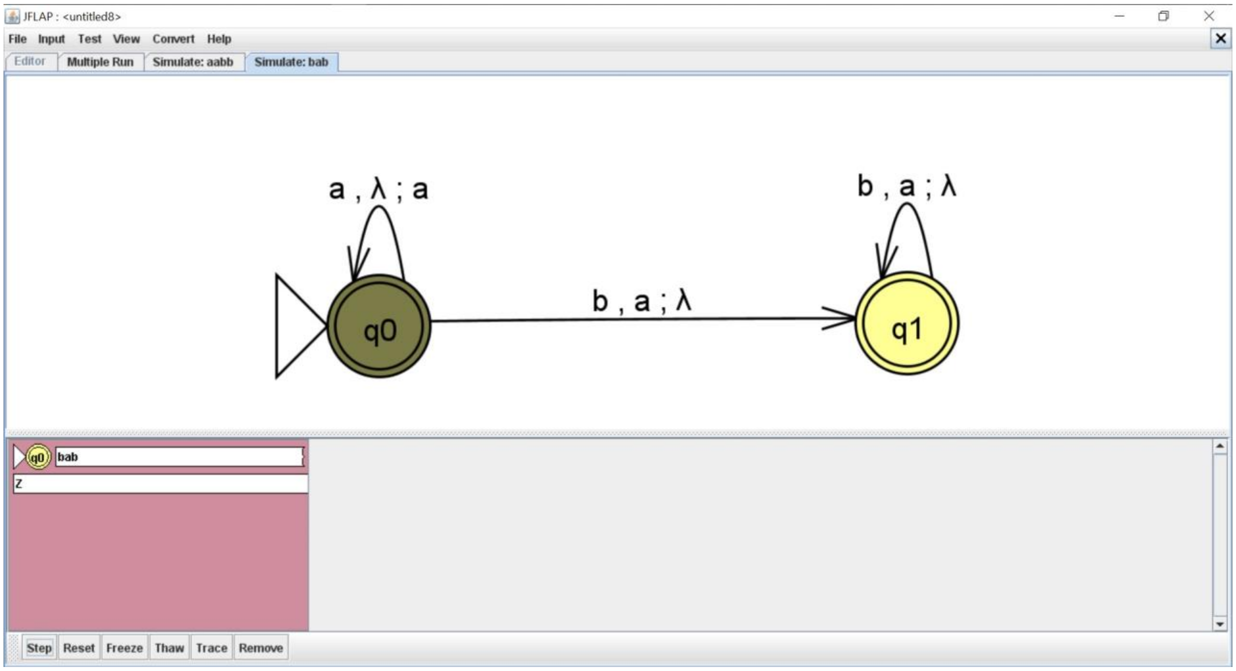
Ashish Kumar Shukla ***1BI20CS033***

Rejecting String: bab



q0 bab

Z



q0 bab

Z