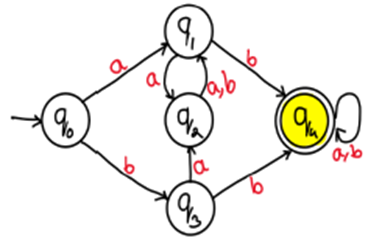
**PROBLEM STATEMENT 9:**

Demonstrate the usage of JFLAP tool for the minimization of the given DFA.

DFA:



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design and minimization of the given DFA.

**Steps:**

Step 1: Calculate the K-equivalent sets of the given DFA

Step 2: Open JFLAP, Click on "File" > "New" > "Finite Automaton."

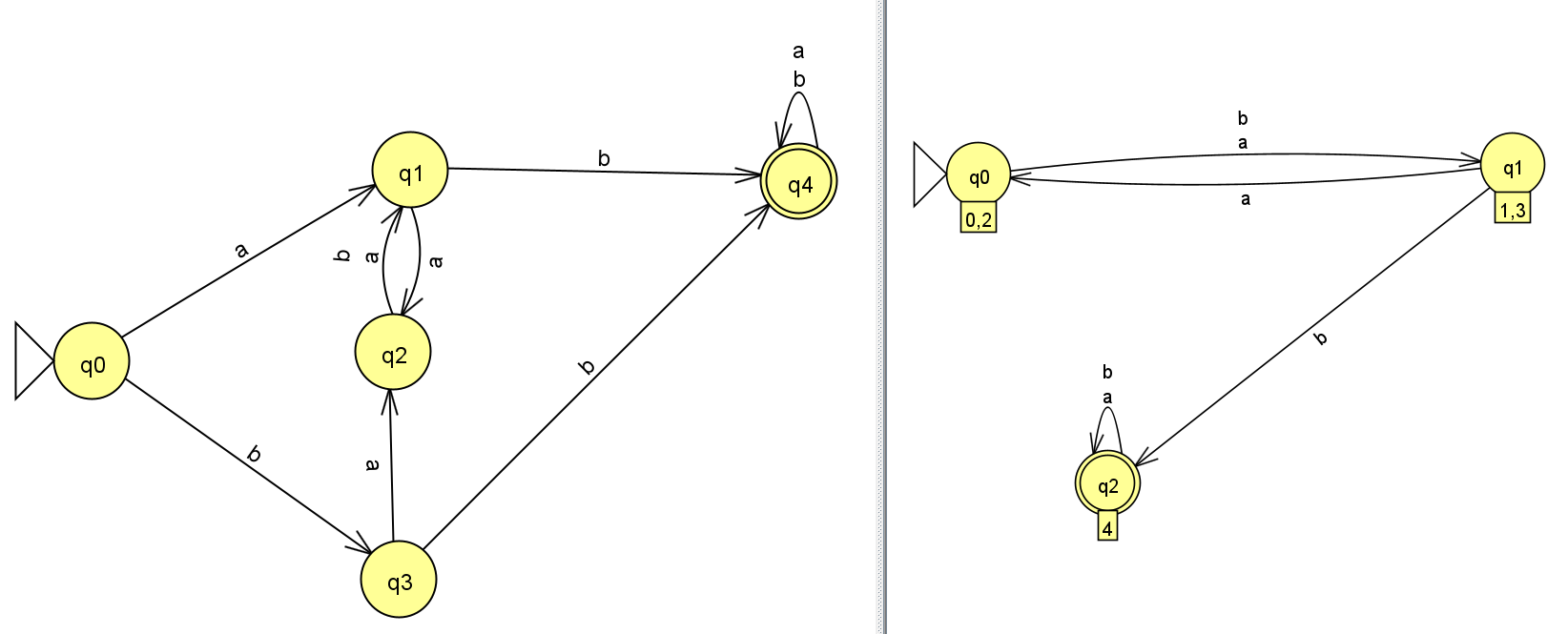
Step 3: In the JFLAP interface, we have a blank canvas.

Step 4: We will design the given DFA on the canvas.

Step 5: Select "Convert" from the menu at the top and choose " Minimize DFA."

Step 6: JFLAP will display the minimized DFA to inspect. It will show the equivalent DFA with fewer states.

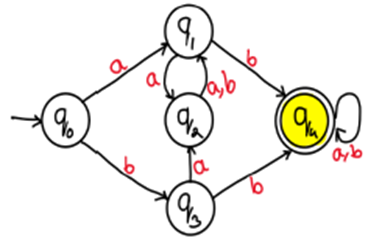
**Output:**



**Problem Statement 10:**

Demonstrate the usage of JFLAP tool to design and convert the following DFA to its grammar

**DFA:**



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design and conversion of the given DFA into its grammar.

**Steps:**

Step 1: Launch the JFLAP tool on your computer.

Step 2: Create a New Project. Click on "File" > "New" > "Automaton" to initiate a new project. Select "DFA" as the automaton type.

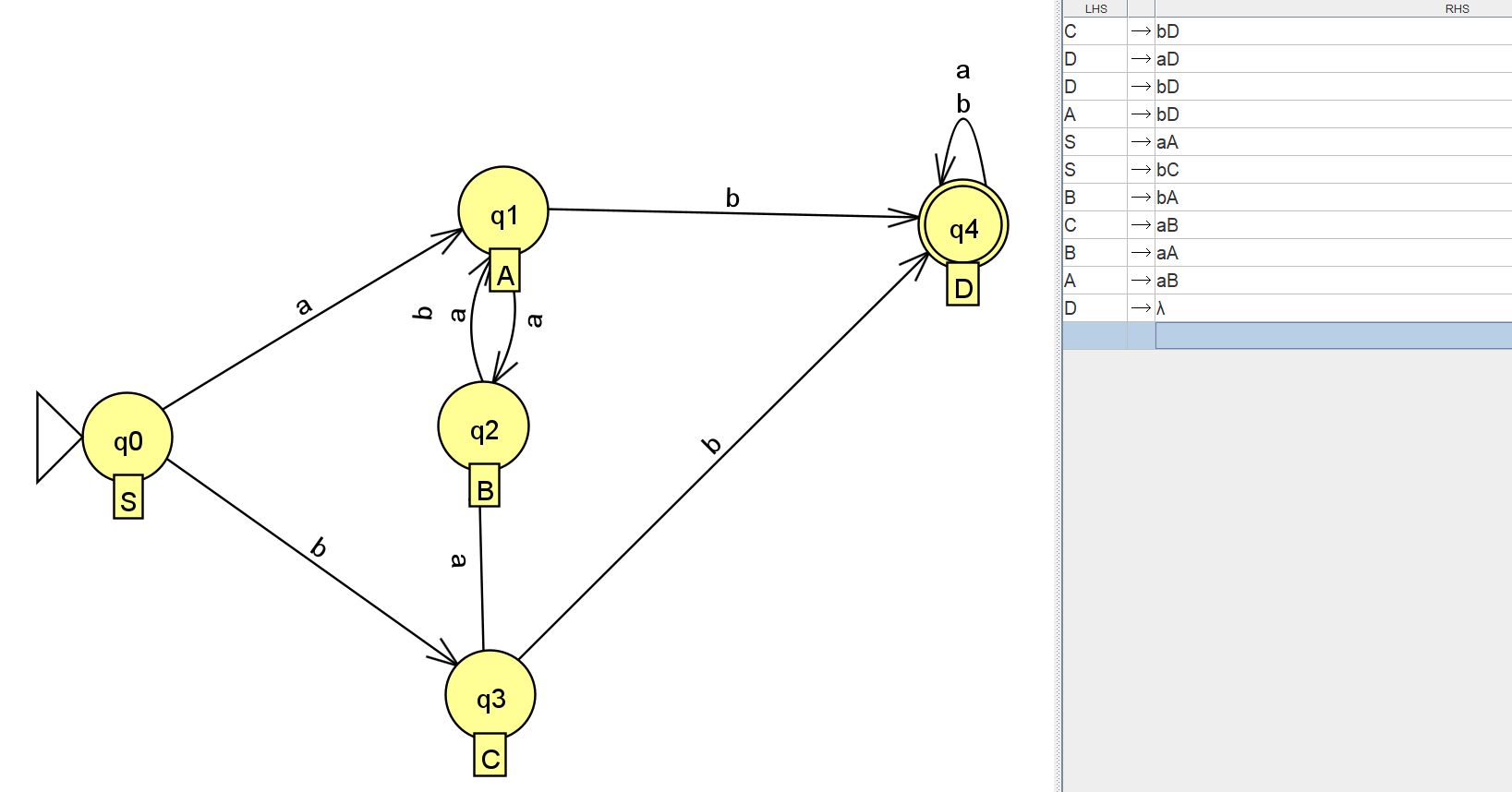
Step 3: Import DFA. If the DFA diagram is already given, you can import it into JFLAP. Go to "File" > "Import" and select the file containing the DFA.

Step 4: Simulate the DFA within JFLAP to verify its correct functionality. Test its acceptance and rejection patterns using various input strings.

Step 5: Convert to Grammar. Navigate to the relevant tool or menu in JFLAP that facilitates the conversion of DFA to grammar. Follow the prompts or options to execute the conversion.

Step 6: Review Grammar Representation. Examine the generated grammar representation corresponding to the given DFA. Understand the grammar rules and symbols that capture the language recognized by the DFA.

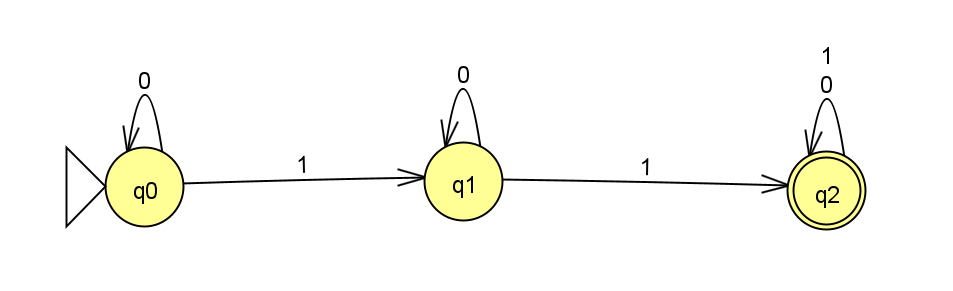
**Output:**



**Problem Statement 11:**

Demonstrate the usages of JFLAP tool to design and generate regular expression the following Deterministic Finite Automata (DFA).

**DFA:**



**DESCRIPTION OF THE PROBLEM STATEMENT:**

JFLAP, a popular tool in formal languages and automata theory, allows users to visually design and manipulate finite automata, including DFAs. In this scenario, we leverage JFLAP to design a DFA and then generate a regular expression that represents the language recognized by the DFA.

**STEPS:**

Step 1: Open JFLAP on your computer.

Step 2: Click on "File" in the menu and select "New" and then "Automaton" to create a new automaton.

Step 3:Define states for the DFA by clicking on the "State" tool and creating states on the canvas.

Define the alphabet using the "Input" tool to enter symbols.

Step 4:Use the "Transition" tool to define transitions between states based on the DFA's transition function. Specify transition labels by indicating the input symbols for each transition.

Step 5: Mark Initial and Final States. Mark the initial state with the "Initial State" tool. Identify final states using the "Final State" tool.

Step 6: Test the DFA. Use the "Input" tool to enter strings and test the DFA. Observe whether the strings are accepted or rejected.

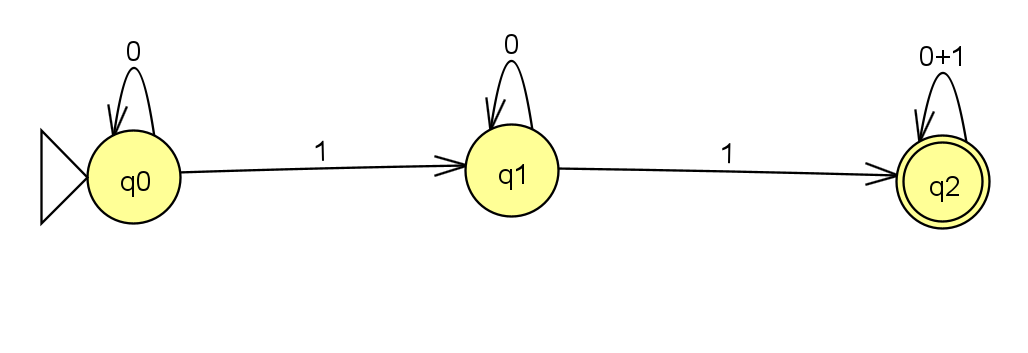
Step 7: Click on "Convert" in the menu, then select "Convert to Regular Grammar."

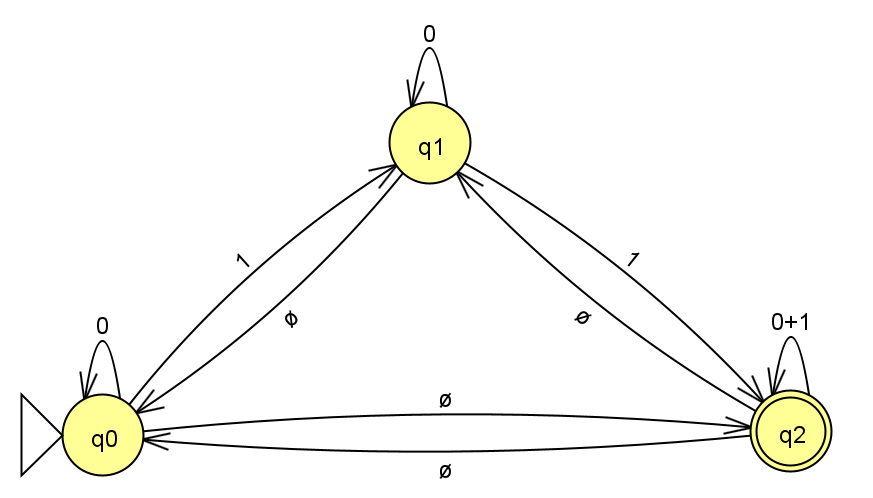
Follow the prompts to generate a regular grammar for the DFA.

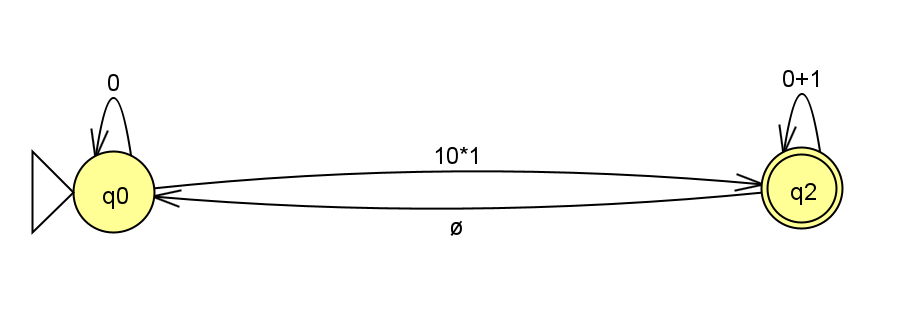
Step 8: Examine the generated regular expression.

Manually refine the regular expression if necessary for simplicity or specific requirements.

**OUTPUT**









**Problem Statement 12:**

Demonstrate the usage of JFLAP tool to design and convert the DFA to equivalent Right-Linear Grammar for the following Language:

L={aw |w € ∑={a,b}

**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design DFA and convert the DFA to equivalent Right-Linear Grammar.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Finite Automaton."

Step 2: In the JFLAP interface, we have a blank canvas.

Step 3: We will design the given DFA on the canvas.

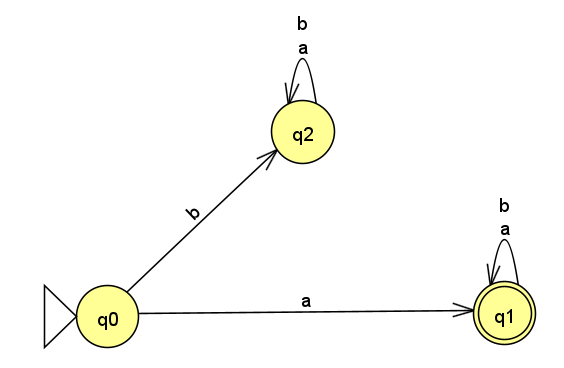
Step 4: Select "Convert" from the menu at the top and choose "Convert To Grammar".

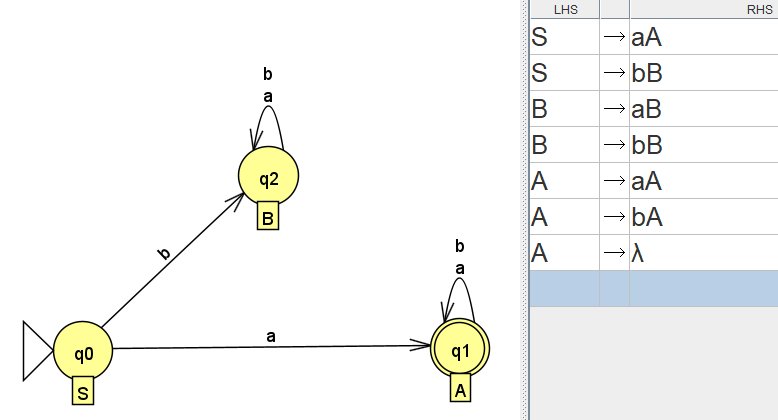
Step 5: Click on each transition and for each transition the corresponding production rule is generated by JFLAP.

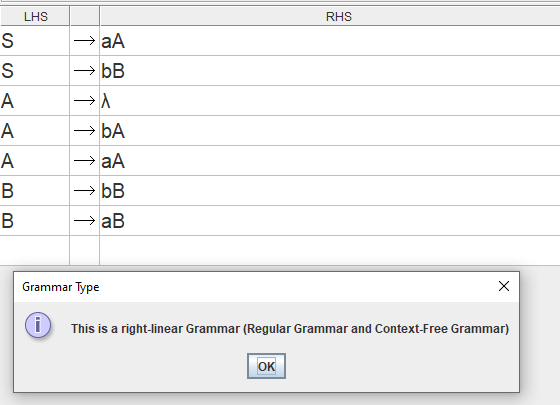
STOP

**Output:**

**DFA Designed:**



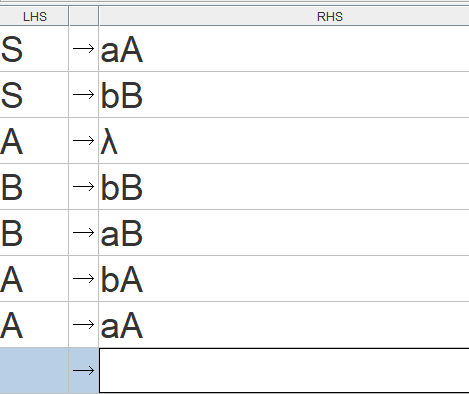




**Problem Statement 13:**

Demonstrate the usage of JFLAP tool to convert the given Right-Linear Grammar to equivalent DFA.

**Given Production Rule:**



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to convert the given Right-Linear Grammar to equivalent DFA.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Grammar"

Step 2: In the JFLAP interface, we have a blank table.

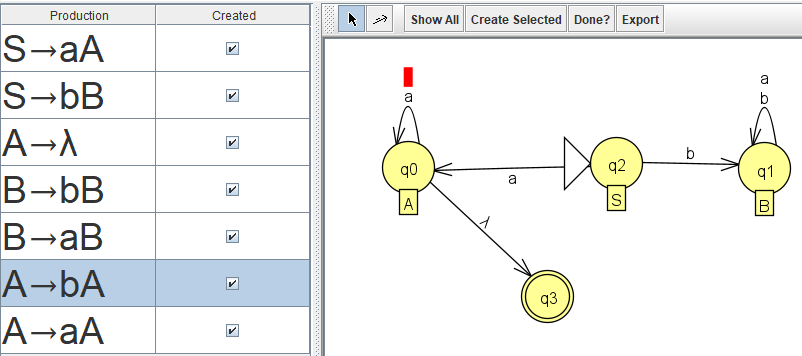
Step 3: We will fill the production rule on the table.

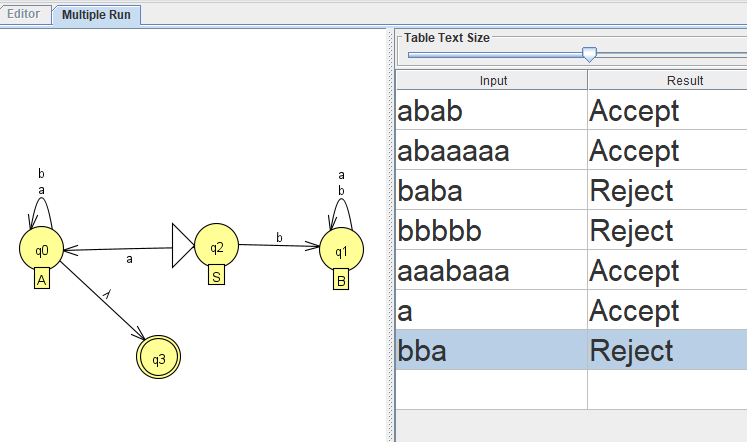
Step 4: Select "Convert" from the menu at the top and choose "Convert Right Linear Grammar To Finite Automata ".

Step 5: JFLAP will generate states according to the production rule and for each production rule, click on the “Create Selected” option on the canvas. This will generate the DFA.

STOP

**OUTPUT:**

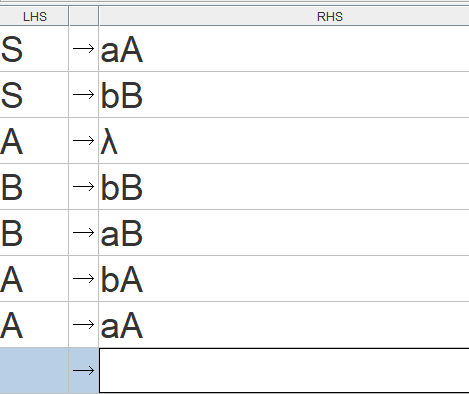




**Problem Statement 14:**

Demonstrate the usage of JFLAP tool to generate a derivation tree for a particular input string and given Right-Linear Grammar.

**Given Production Rule:**



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to generate a derivation tree for a particular input string and given Right-Linear Grammar.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Grammar"

Step 2: In the JFLAP interface, we have a blank table.

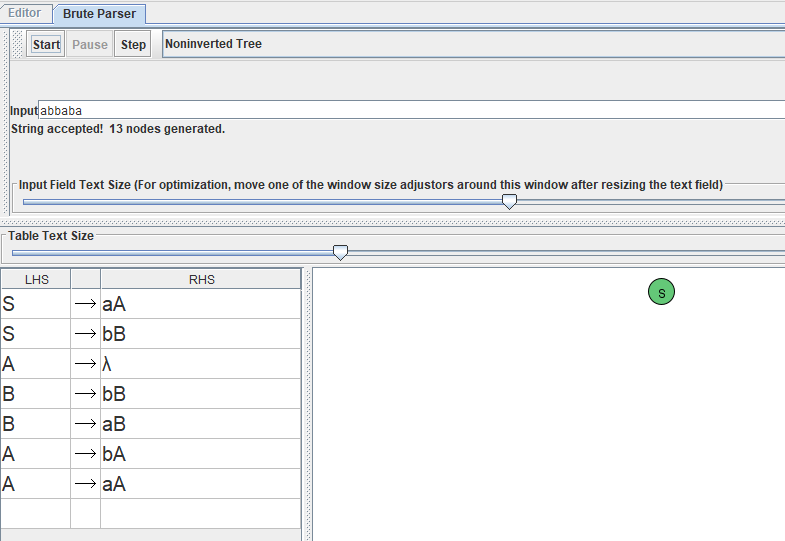
Step 3: We will fill the production rule on the table.

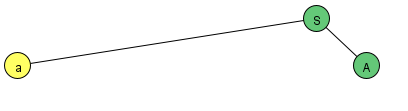
Step 4: Select "Input" from the menu at the top and choose "Brute Force Parse".

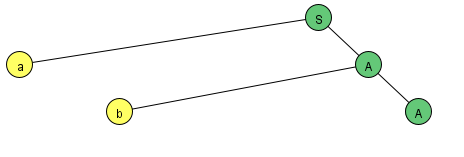
Step 5: JFLAP will generate a new window where the input string is entered and for each character in the input string derivation tree is generated by JFLAP by clicking on the “Start” option for the first and “Step” for other characters.

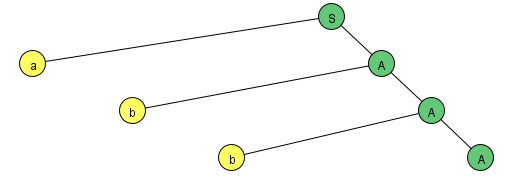
STOP

**Output:**

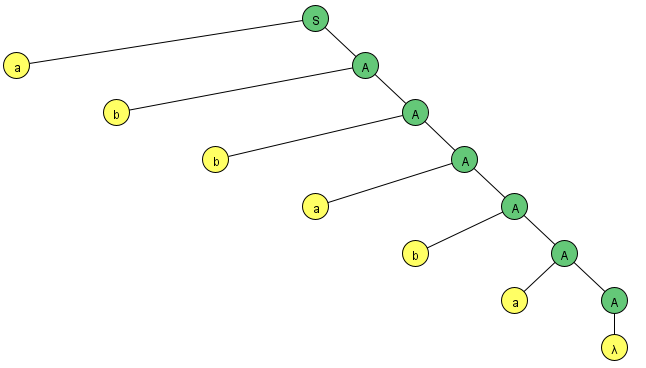








and so on…





**Problem Statement 15:**

Design and simulate Moore Machine in JFLAP to find 1’s complement of a binary string.

**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design and simulate Moore Machine in JFLAP to find 1’s complement of a binary string.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Moore Machine"

Step 2: In the JFLAP interface, we have a blank table.

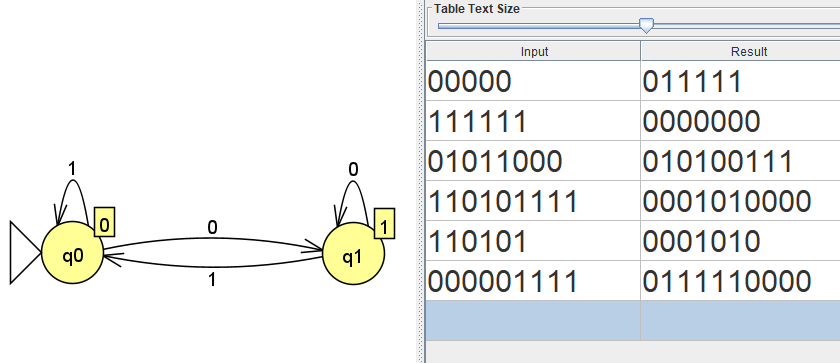
Step 3: We will create states and transitions based on the Outputs.

Step 4: Select "Input" from the menu at the top and choose "Multiple Run" to simulate the Moore Machine.

Step 5: On the right side enter the binary strings to find the 1’s complement and then click on “Run Inputs” from the right-bottom side.

STOP

**Output:**



**Problem Statement 16:**

Design and simulate Mealy Machine in JFLAP to find 1’s complement of a binary string.

**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design and simulate Mealy Machine in JFLAP to find 1’s complement of a binary string.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Moore Machine"

Step 2: In the JFLAP interface, we have a blank table.

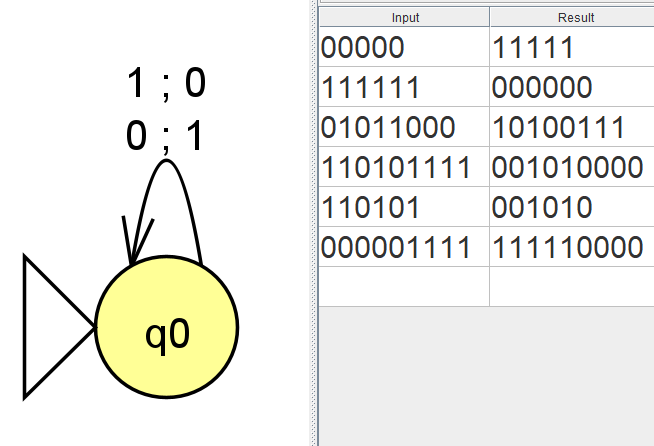
Step 3: We will create states and transitions based on the Outputs.

Step 4: Select "Input" from the menu at the top and choose "Multiple Run" to simulate the Moore Machine.

Step 5: On the right side enter the binary strings to find the 1’s complement and then click on “Run Inputs” from the right-bottom side.

STOP

**Output:**



**Problem Statement 17:**

Demonstrate the usage of JFLAP tool to simulate PushDown Automata (PDA), for the following language L={anbn | n>=1, ∑={a,b}}.

**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to design and simulate simulate PushDown Automata (PDA), for the given language.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > " Pushdown Automaton"

Step 2: In the JFLAP interface, we have a blank table.

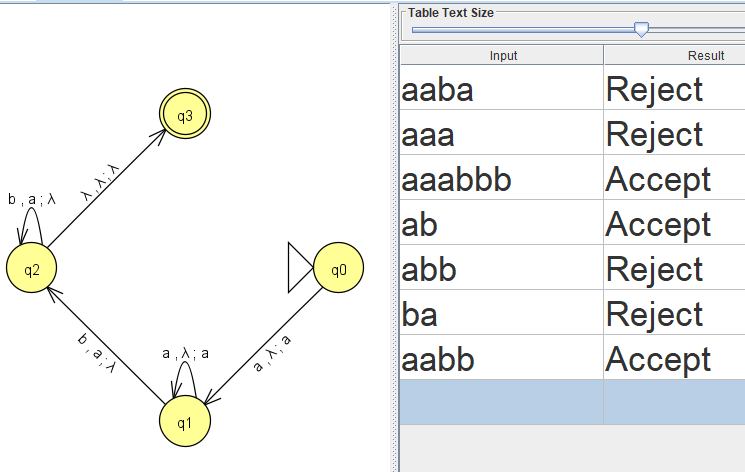
Step 3: We will create states, transitions, and a stack to keep track of the 'a's and 'b's.

Step 4: Select "Input" from the menu at the top and choose "Multiple Run" to simulate the Moore Machine.

Step 5: On the right side enter the input strings to test the PDA and then click on “Run Inputs” from the right-bottom side.

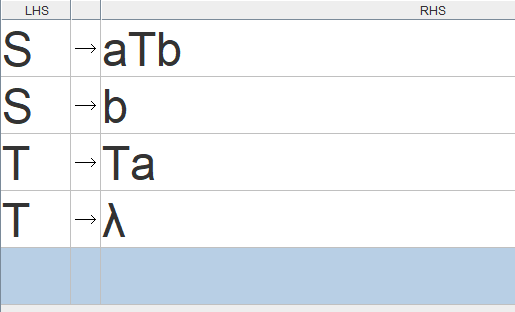
STOP

**Output:**



**Problem Statement 18:**

Demonstrate the usage of JFLAP tool to convert the following Context Free Grammar to NPDA, G=(VN, VT, S, P) where set of productions are given as:



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to convert the following Context Free Grammar to NPDA.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Grammar"

Step 2: In the JFLAP interface, we have a blank table.

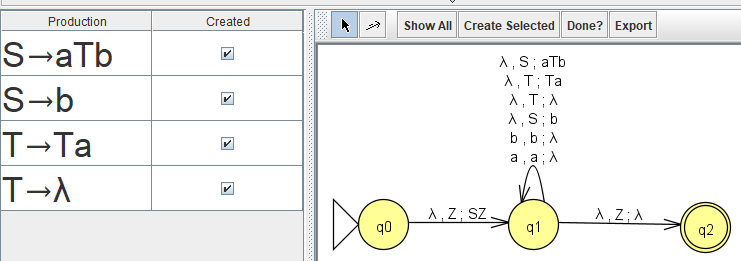
Step 3: We will fill the production rule on the table.

Step 4: Select "convert" from the menu at the top and choose "CFG to PDA(LL)".

Step 5: JFLAP will generate a new window where the NPDA is displayed.

STOP

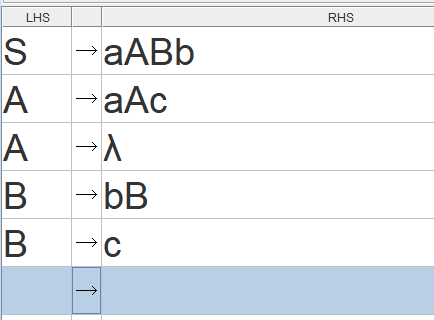
**Output:**



**Problem Statement 19:**

Demonstrate the usages of JFLAP tool to simulate LL (1) top-down parser.

**Given Production Rule:**



**Description of the problem statement:**

JFLAP (Java Formal Language Automata Package) is a popular java tool for designing and simulating finite automata and other formal languages. This demonstrates how to use JFLAP to to simulate LL (1) top-down parser.

**Steps:**

START

Step 1: Open JFLAP, Click on "File" > "New" > "Grammar"

Step 2: In the JFLAP interface, we have a blank table.

Step 3: We will fill the production rule on the table.

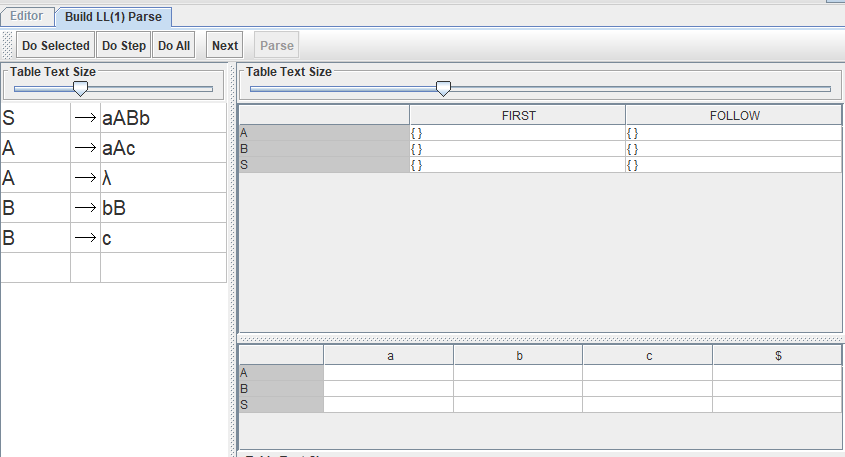
Step 4: Click on “Input” and then “Build LL(1) Parse Table".

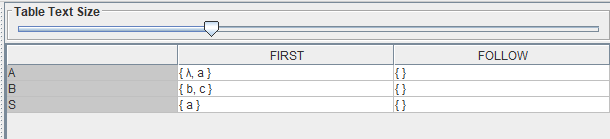
Step 5: Now fill the first, follow sets and LL(1) Parse Table for the given production rules.

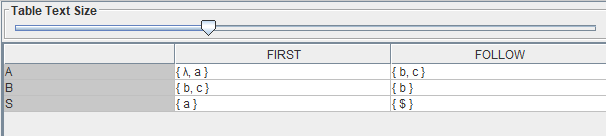
Step 6: Click on “Parse” option and then enter a test string and then simulate the string.

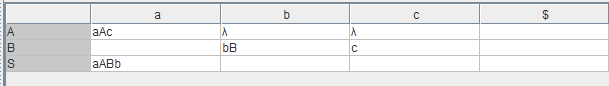
STOP

**Output:**









Parsing:

For String: aaaacccbbcb

