PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No. 6 (A)

Aim : Using JFLAP, create a DFA from a given regular expression.

Objective: Simulate a program to implement even number of Zero's in String

a. Using Jflap tool
b. Using finite automata

Outcome: Students are able implement even number of Zero's in String.

Theory:

What is JFLAP: -

JFLAP program makes it possible to create and simulate automata. Learning about automata with pen and paper can be difficult, time consuming and error-prone. With JFLAP we can create automata of different types and it is easy to change them as we want. JFLAP supports creation of DFA and NFA, Regular Expressions, PDA, Turing Machines, Grammars and more.

Setup: -

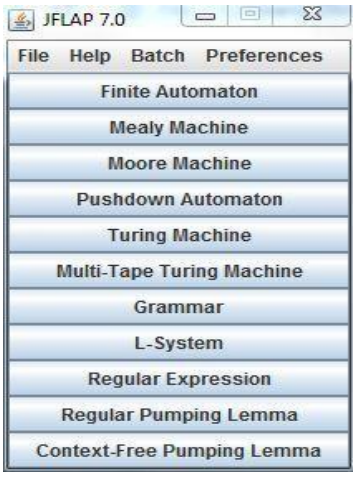
JFLAP is available from the homepage: (www.JFLAP.org). From there press "Get FLAP" and follow theinstructions. You will notice that JFLAP have a .JAR extension. This means that you need Java to run JFLAP. With Java correctly installed you can simply select the program to run it. You can also use a command console run it from the files current directory with, Java –jar JFLAP.jar.

Using JFLAP: -

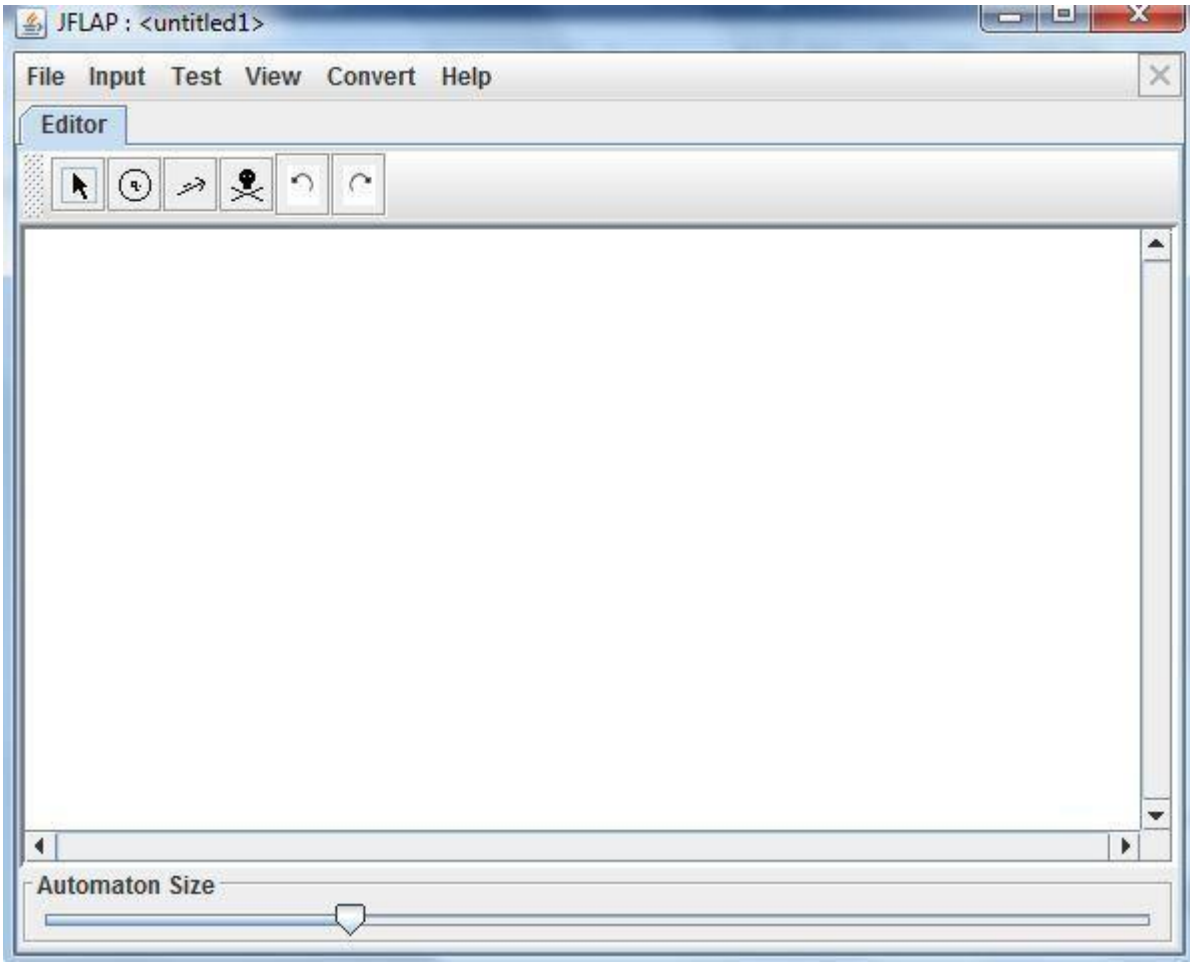
When you first start JFLAP you will see a small menu with a selection of eleven different automata and rule sets. Choosing one of them will open the editor where you create chosen type of automata. Usually you can create automata containing states and transitions but there is also creation of Grammar and Regular Expression which is made with a text editor.

DFA from a given regular expression: -

First, we need to select Regular Expression from the JFLAP Menu.


JFLAP 7.0
File
Help
Batch
Preferences
Finite Automaton
Mealy Machine
Moore Machine
Pushdown Automaton
Turing Machine
Multi-Tape Turing Machine
Grammar
L-System
Regular Expression
Regular Pumping Lemma
Context-Free Pumping Lemma

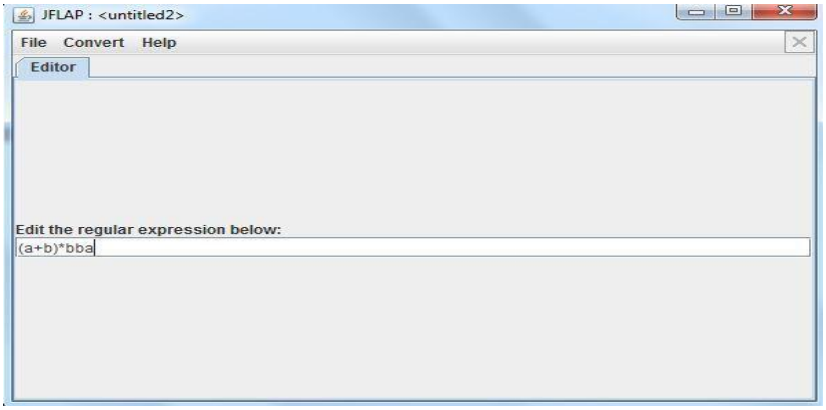
Now you should have an empty window in front of you. You will have a couple of tools and features at your disposal.



The toolbar contains six tools, which are used to edit automata.

Attribute Editor Tool, changes properties and position of existing states and transitions.
State Creator Tool, creates new states.
Transition Creator Tool, creates transitions.
Deletion Tool, deletes states and transitions.
Undo/Redo, changes the selected object prior to their history.

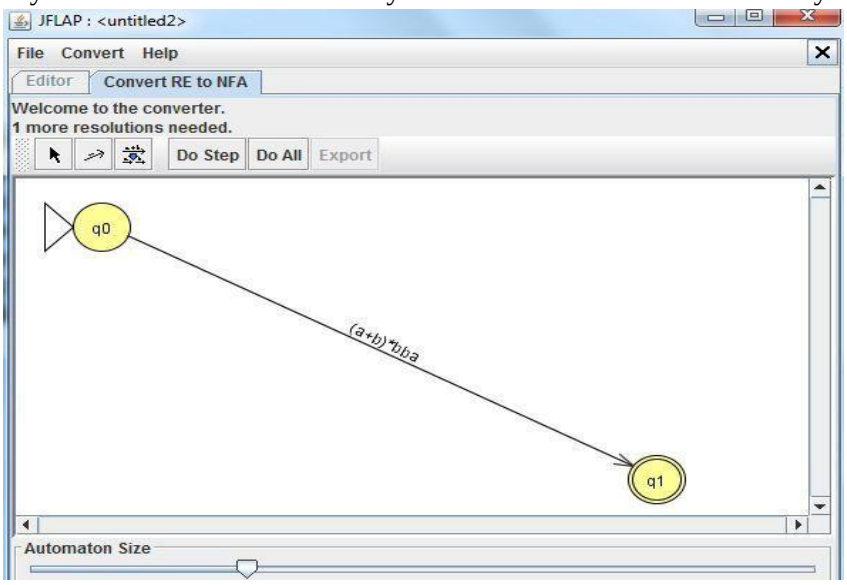
Regular Expressions can be typed into JFLAP to be converted to an NFA



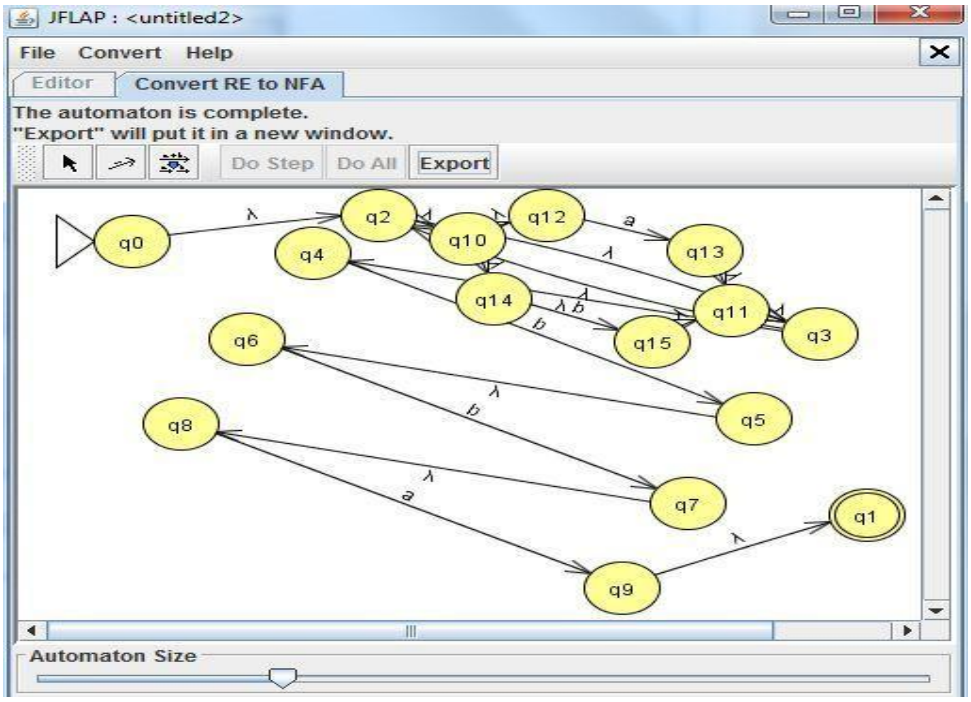
Choose Regular Expression in the main menu, then just type the expression in the textbox. Definitions for Regular Expressions in JFLAP:

- **Kleene Star**
- **+ Union**
- **! Empty String**

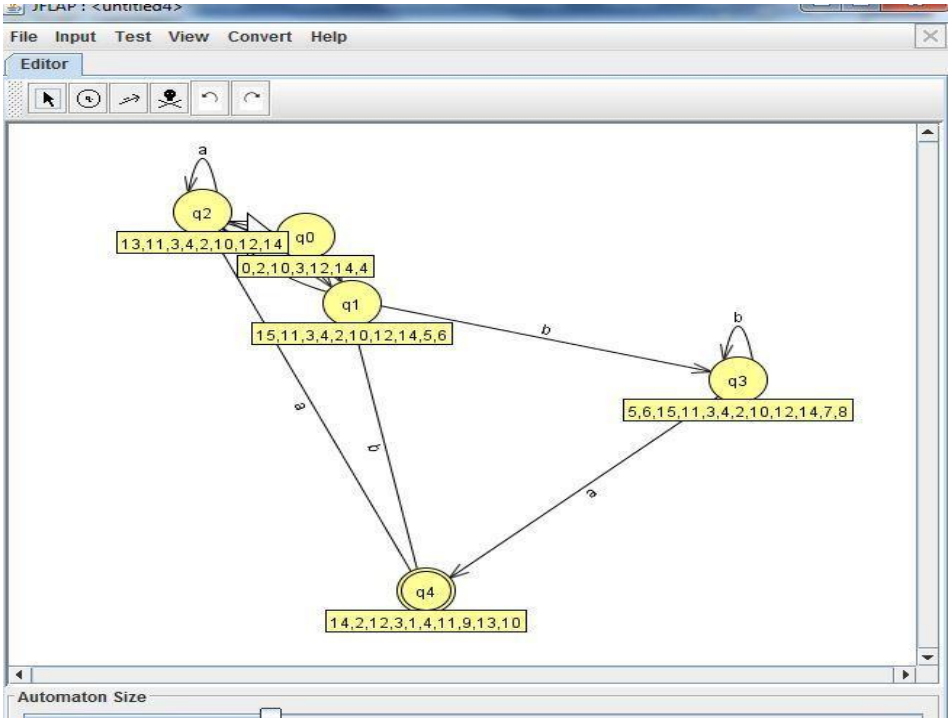
Correctly written expressions can then be converted to an NFA. To convert your expression select Convert → Convert to NFA. The conversion will begin with two states and a transition with your RegularExpression. With the (D)e-expressionify Transition tool you can break down the Regular Expression intosmaller parts. Each transition will contain a sub expression. The next step is to link every rule with lambda transitions. Add new transition between states that should be connected with the Transition Tool. If you are unsure what to do you can select Do Step to automatically make the next step. If you want the NFA immediately Do All creates the whole NFA for you.



You can notice how the conversion differs depending on how the Regular Expression looks. For example the expression a+b results in a fork, were either ‘a’ or ‘b’ can be chosen.



Now finally convernt NFA to DFA:-



PART B

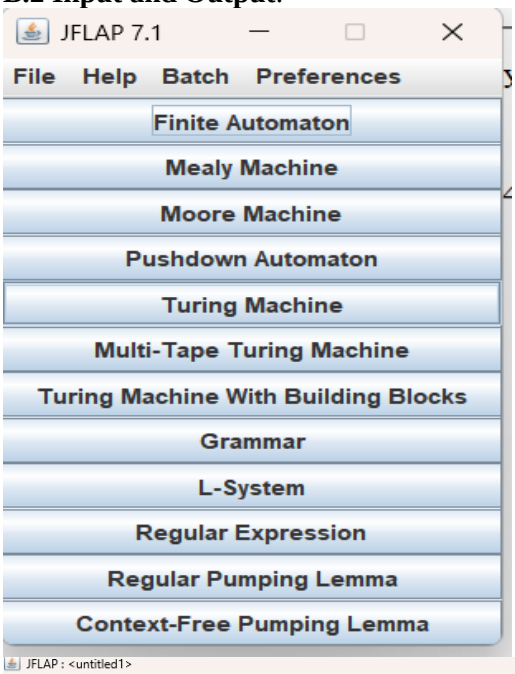
(PART B : TO BE COMPLETED BY STUDENTS)

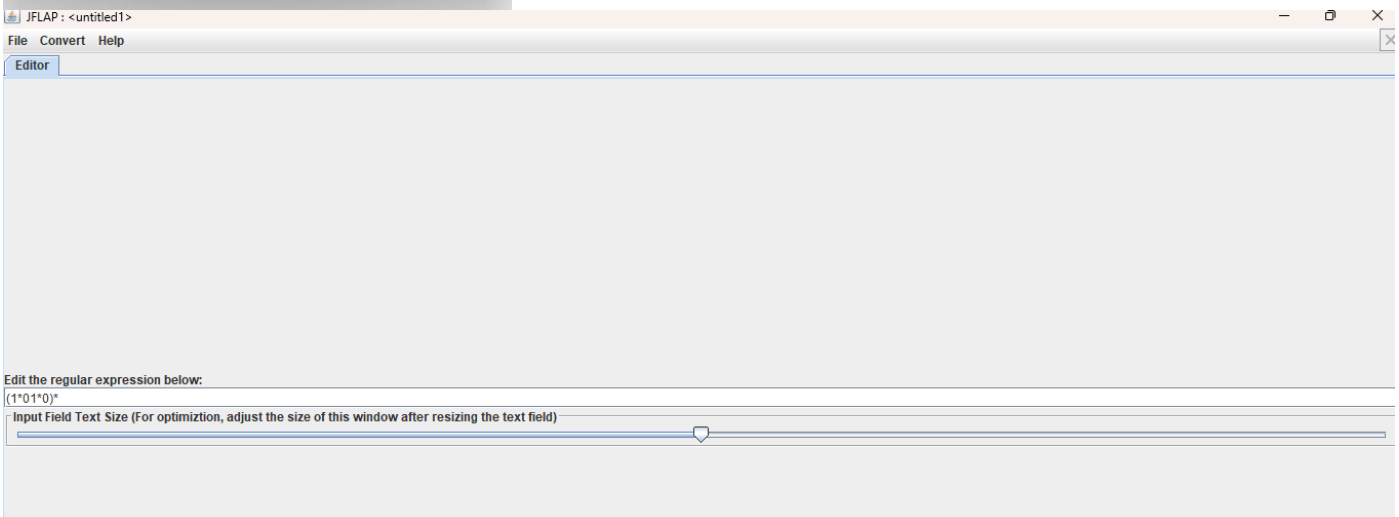
(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded at the end of the practical)

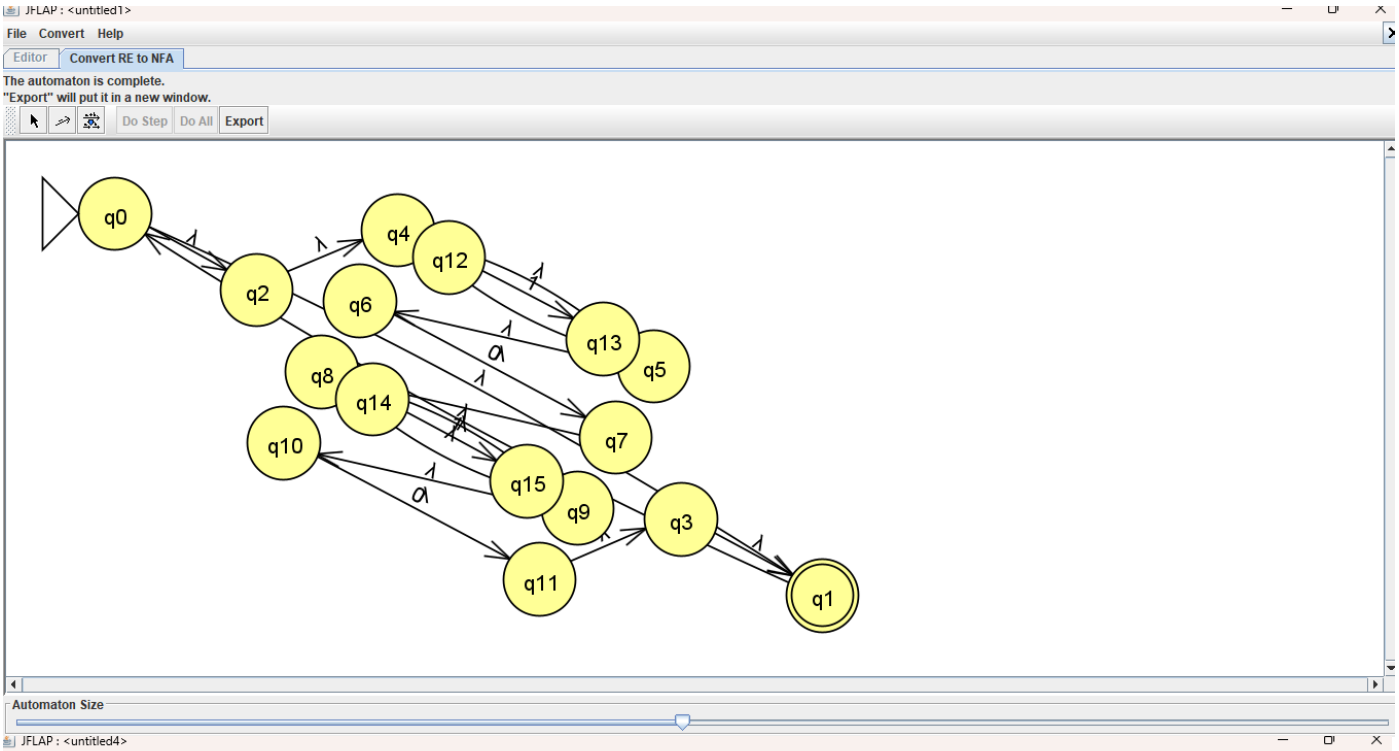
Roll. No. B48	Name: Aryan Unhale
Class: TE B COMPS	Batch:B3
Date of Experiment:18/3/25	Date of Submission:4/4/25
Grade:	

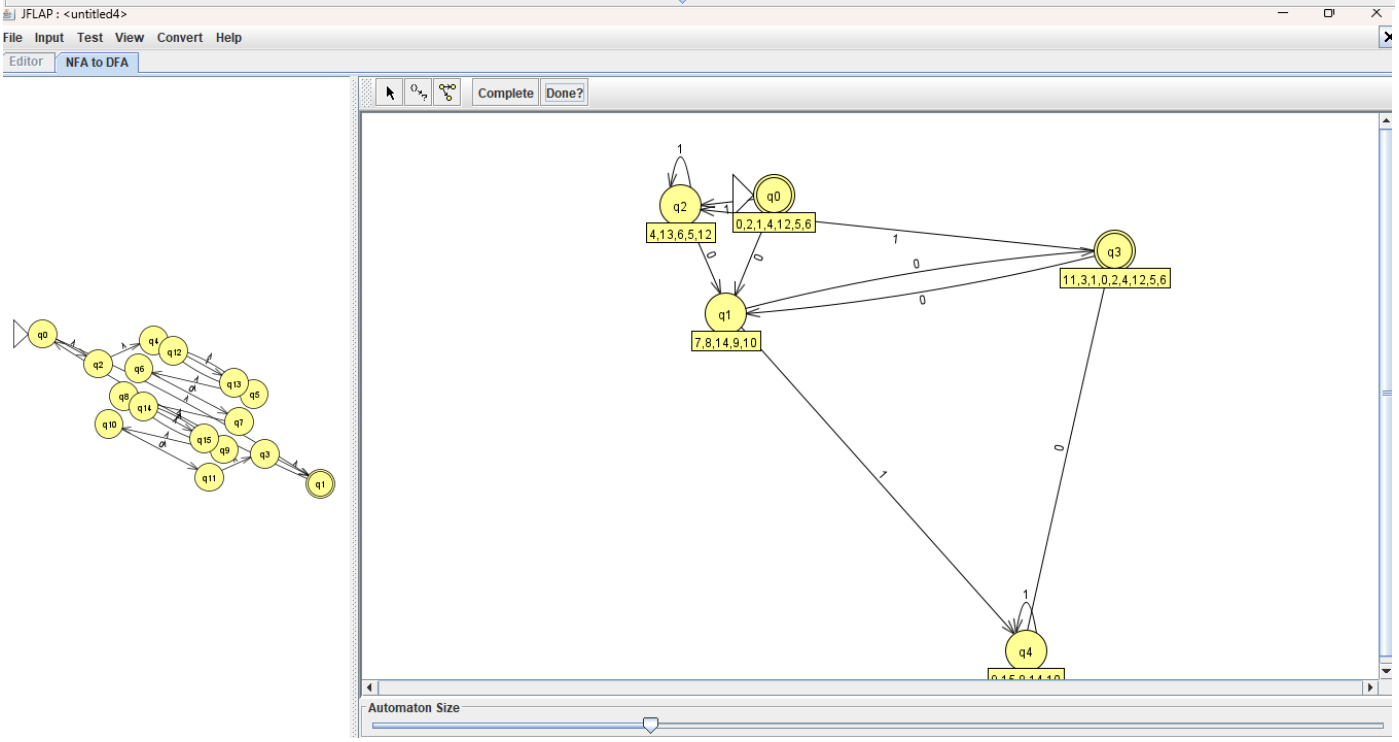
B.1 Software Code written by student:

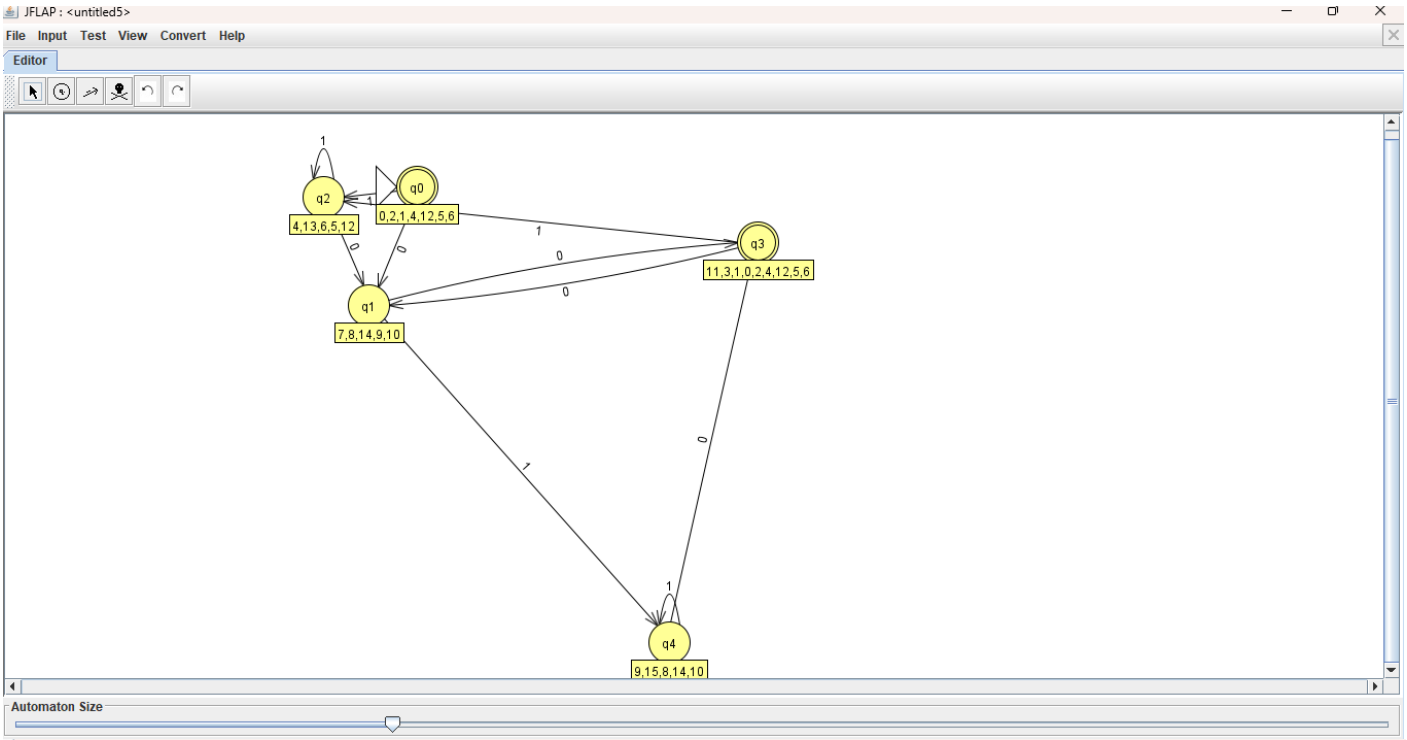
B.2 Input and Output:

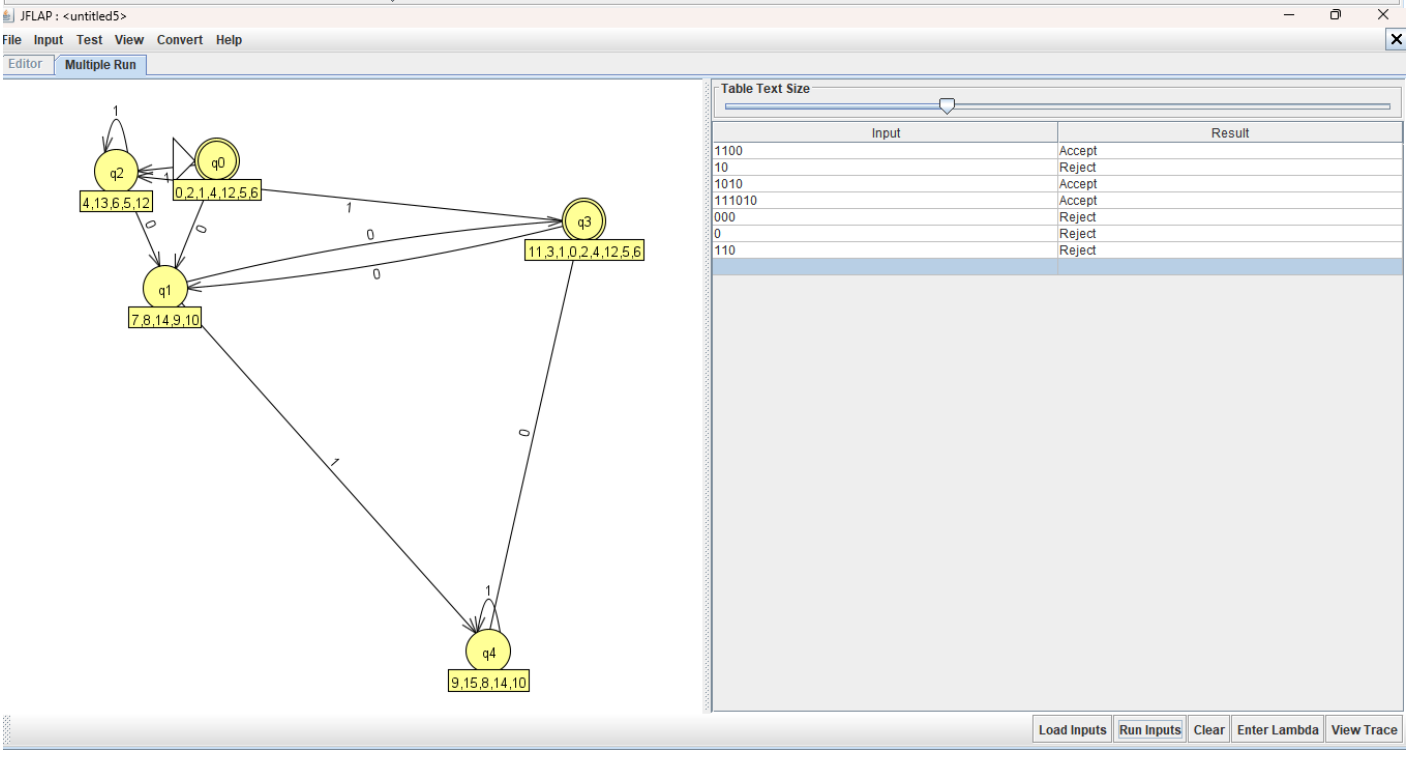





JFLAP : <untitled1>
File Convert Help
Editor Convert RE to NFA
The automaton is complete.
"Export" will put it in a new window.
Do Step Do All Export
q0 q2 q4 q12 q6 q13 q5 q8 q14 q10 q7 q15 q9 q3 q11 q1
λ
a
Automaton Size


JFLAP : <untitled4>
File Input Test View Convert Help
Editor NFA to DFA
Complete Done?
q2 4,13,6,5,12
q0 0,2,1,4,12,5,6
q3 11,3,1,0,2,4,12,5,6
q1 7,8,14,9,10
q4
1
0
Automaton Size


JFLAP : <untitled5>
File Input Test View Convert Help
Editor
q0
0,2,1,4,12,5,6
q1
7,8,14,9,10
q2
4,13,6,5,12
q3
11,3,1,0,2,4,12,5,6
q4
9,15,8,14,10
Automaton Size


JFLAP : <untitled5>
File Input Test View Convert Help
Editor
Multiple Run
Table Text Size
Input
Result
1100 Accept
10 Reject
1010 Accept
111010 Accept
000 Reject
0 Reject
110 Reject
Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace

B.3 Observations and learning:

- The DFA correctly accepts strings with even occurrences of 0's while ignoring the number of 1's.
- The conversion process from Regular Expression → NFA → DFA was automated using JFLAP.
- Any input string with an odd number of 0's was rejected, confirming the correctness of the DFA.
- JFLAP provided step-by-step simulation to visualize state transitions.

B.4 Conclusion:

The experiment successfully demonstrated the use of JFLAP to create a DFA for recognizing strings with an even number of 0's.
The conversion from Regular Expression to NFA and then to DFA was efficiently done using JFLAP.
This experiment reinforced the concept of Finite Automata and Regular Expressions and their role in language recognition.
JFLAP proved to be an effective tool for designing and testing automata with ease.

B.5 Question of Curiosity

1. How does JFLAP tool help in designing finite automata?

Ans :

JFLAP (Java Formal Language and Automata Package) is a software tool designed to assist users in experimenting with formal languages and automata. It provides a range of functionalities, and when it comes to designing finite automata, JFLAP offers several benefits:

1. Graphical Interface:
JFLAP provides a user-friendly graphical interface that allows users to visually design and manipulate finite automata. This visual representation makes it easier for users to understand and work with the different components of an automaton, including states, transitions, and final states.

2. Automaton Construction:
Users can create finite automata from scratch by adding states and defining transitions between them. JFLAP supports various types of automata, including Deterministic Finite Automata (DFA), NonDeterministic Finite Automata (NFA), Pushdown Automata (PDA), and more.

3. Transition Editing:
The tool allows users to define and edit transitions between states easily. Users can specify input symbols that trigger transitions, and JFLAP provides an intuitive interface for managing

these transitions.

4. Simulations and Testing:
JFLAP enables users to simulate the behavior of their designed finite automata. This simulation feature allows users to input strings and observe how the automaton processes the input, helping in verifying the correctness of the designed automaton.

5. Conversion Between Automata Types:
JFLAP supports the conversion between different types of automata. For example, users can convert a regular expression to a finite automaton or transform an NFA to a DFA. This flexibility is valuable in exploring the relationships between different forms of automata.

6. Algorithm Visualization:
The tool includes features to visualize the execution of various algorithms related to automata, such as the conversion algorithms or the process of recognizing a language. This visual representation aids in understanding the theoretical concepts behind the algorithms.

7. Error Detection and Debugging:
JFLAP assists users in identifying errors in their automaton designs. If there are issues with the structure or logic of the automaton, the tool can provide error messages or highlight problematic areas, helping users to debug their designs.

8. Educational Tool:
JFLAP is commonly used in educational settings to teach formal languages and automata theory. It allows students to interactively explore and experiment with different automata concepts, reinforcing theoretical knowledge through practical application.

9. Export and Documentation:
Users can export their designed automata in various formats, making it easy to share and document their work. This feature is helpful for creating educational materials, presentations, or documentation of automaton designs.

2. How JFLAP can be used to design programming language?

Ans:

JFLAP is primarily designed for experimenting with formal languages and automata, and while it's a valuable tool for illustrating concepts related to programming languages, it is not specifically
tailored for designing entire programming languages. However, it can aid in visualizing certain aspects of programming language design. Here's how JFLAP can be used in the context of designing programming languages:

1. Regular Expressions:
JFLAP allows users to design finite automata and regular expressions. In the context of programming languages, regular expressions are often used to define the syntax of tokens, such as keywords, identifiers, and literals. You can use JFLAP to create a finite automaton that recognizes these token patterns.

2. Lexical Analysis:
Programming languages often go through a lexical analysis phase to break down the source code into tokens. JFLAP can be used to design finite automata or regular expressions for the lexical rules of a language. This includes specifying patterns for keywords, operators, and other lexical elements.

3. Automata for Language Recognition:
You can use JFLAP to design automata that recognize the syntactic structure of a programming language. Finite automata, pushdown automata, or context-free grammars can be created to model the language's grammar rules and syntax.

4. Transition Diagrams:
JFLAP allows the creation of transition diagrams for finite automata and other types of automata. These diagrams can be used to visually represent the state transitions in the language's syntax, making it easier to understand the flow of control in the language.

5. Simulation and Testing:
JFLAP provides simulation features that allow users to test their automata with different inputs. This can be useful for verifying that the designed automata correctly recognize or reject valid and
invalid language constructs.

6. Understanding Language Constructs:
By using JFLAP to design automata for specific language constructs, you can gain a deeper understanding of how the language processes different elements. For instance, you can model the parsing of expressions, statements, or control flow structures using appropriate automata.

7. Teaching and Learning:
JFLAP is widely used as an educational tool for teaching formal languages and automata theory. In the context of programming languages, it can help students visualize and experiment with different language constructs, fostering a better understanding of language design principles.

3. Can we convert every DFA to NFA?
Ans:

Yes, it is indeed possible to convert every Deterministic Finite Automaton (DFA) to a Non Deterministic Finite Automaton (NFA). The conversion process involves creating an equivalent NFA that recognizes the same language as the original DFA. How to convert a DFA to an NFA:

1. State Representation:
Each state in the DFA becomes a state in the NFA.

2. Transition Function:
For each transition in the DFA, create an equivalent set of transitions in the NFA.

If the DFA transitions from state A to state B on input symbol 'a', the NFA will have a set of transitions from state A to state B on input symbol 'a'.

3. Start State:
The start state of the NFA is the same as the start state of the DFA.

4. Accepting States:
The set of accepting states in the NFA is the same as the set of accepting states in the DFA.