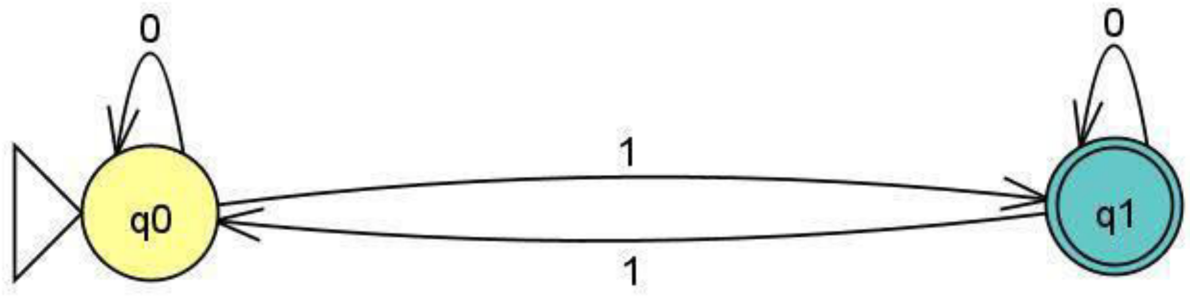
# Experiment-03

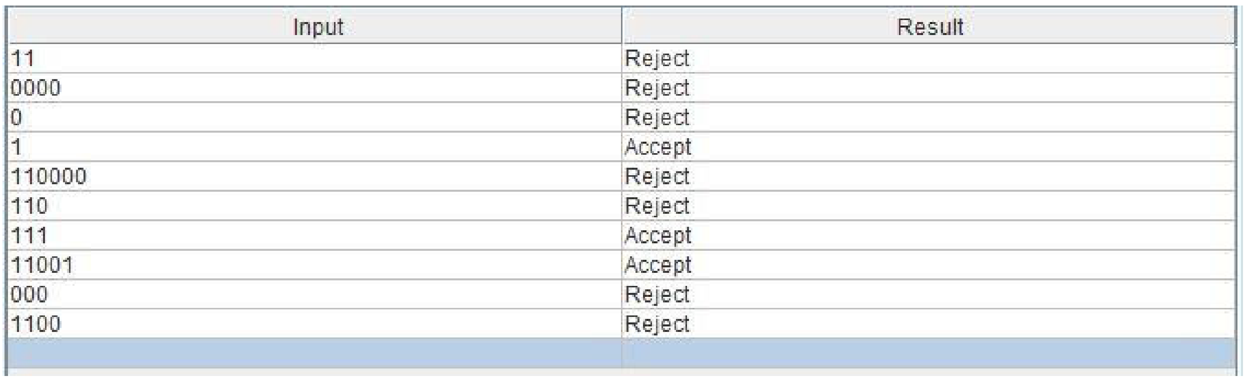
**Aim:** Design a DFA which will accept all the strings containing odd number of 1's over an alphabet (0, 1) and write a program to implement the DFA.

**Code:**

```
#include<iostream>
#include<string>
using namespace std;
int main(){
   string input;
   cout<<"Enter String: ";
   cin>>input;
   int countOnes=0;
   for(char c:input){
      if(c=='1'){
         countOnes++;
      }
   }
   if(countOnes%2!=0){
      cout<<"String accepted"<<endl;
   }
   else{
      cout<<"String not accepted"<<endl;
   }
   return 0;
}
```

**Output:**


$$\begin{tikzcd} \arrow[r] & q0 \arrow[loop above, "0"] \arrow[rr, bend left, "1"] & & q1 \arrow[loop above, "0"] \arrow[ll, bend left, "1"] \end{tikzcd}$$



| Input | Result |
|---|---|
| 11 | Reject |
| 0000 | Reject |
| 0 | Reject |
| 1 | Accept |
| 110000 | Reject |
| 110 | Reject |
| 111 | Accept |
| 11001 | Accept |
| 000 | Reject |
| 1100 | Reject |

# COMPILER DESIGN LAB (CSP353)

**B.TECH 3nd YEAR**

**SEMESTER: 6TH**

**SESSION: 2024-2025**

**Submitted By:**

Ram Ashish Kumar

2021001909

**Section: CSE-A**

**Submitted To:**

Mr. Kanderp Mishra

Associate Professor

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**School of Engineering and Technology.**

**SHARDA UNIVERSITY, GREATER NOIDA.**

# INDEX

| S.NO | EXPERIMENT | DATE OF EXPERIMENT | DATE OF SUBMISSION | SIGNATURE |
|---|---|---|---|---|
| 1. | Introduction of JFLAP a software tool. | | | |
| 2. | Design a DFA which will accept all the strings containing even number of 0's over an alphabet (0, 1) and write a program to implement the DFA. | | | |
| 3. | Design a DFA which will accept all the strings containing odd number of I's over an alphabet (0, 1) and write a program to implement the DFA. | | | |
| 4. | Design a DFA which will accept all the strings ending with 00 over an alphabet (0, 1) and write a program to implement the DFA. | | | |
| 5. | Design a DFA which will accept all the strings containing mod 3 of 0's over an alphabet (0, 1) and write a program to implement the dfa. | | | |
| 6. | Write an algorithm and program to compute FIRST and FOLLOW function. | | | |
| 7. | Write an algorithm and program on Recursive Descent parser. | | | |
| 8. | Write a C program for constructing of L.L. (1) parsing | | | |
| 9. | Write a program to Design LALR Bottom up Parser | | | |
| 10. | Convert The BNF rules into Yace form and write code to generate abstract syntax tree. | | | |
| 11. | To write a C program to implement Symbol Table | | | |

# EXPERIMENT-1

## Q1) Introduction of JFLAP a software tool.

JFLAP (Java Formal Languages and Automata Package) is a powerful software tool used for experimenting, visualizing, and simulating various concepts in formal languages, automata theory, and computational complexity. Developed by Susan Rodger and her team at Duke University, JFLAP provides a user-friendly interface for students, educators, and researchers to explore fundamental topics in theoretical computer science.

**Key features of JFLAP include:**

Automata Simulation: JFLAP allows users to create, manipulate, and simulate finite automata, pushdown automata, Turing machines, and other types of automata. Users can define states, transitions, and input strings to observe the behavior of these machines.

Grammar Analysis: Users can define context-free grammars (CFGs) and regular grammars within JFLAP. The tool offers functionalities for converting between different types of grammars, checking for ambiguity, and generating parse trees.

Turing Machine Editor: JFLAP provides a graphical interface for designing and simulating Turing machines. Users can define the machine's states, transitions, and input alphabet, and observe its execution step-by-step.

Minimization and Conversion: JFLAP offers algorithms for minimizing finite automata, converting between different types of automata (e.g., NFA to DFA), and optimizing grammar representations.

Algorithm Visualization: JFLAP includes visualizations for algorithms such as the CYK parsing algorithm, which can help users understand how these algorithms work through interactive demonstrations.

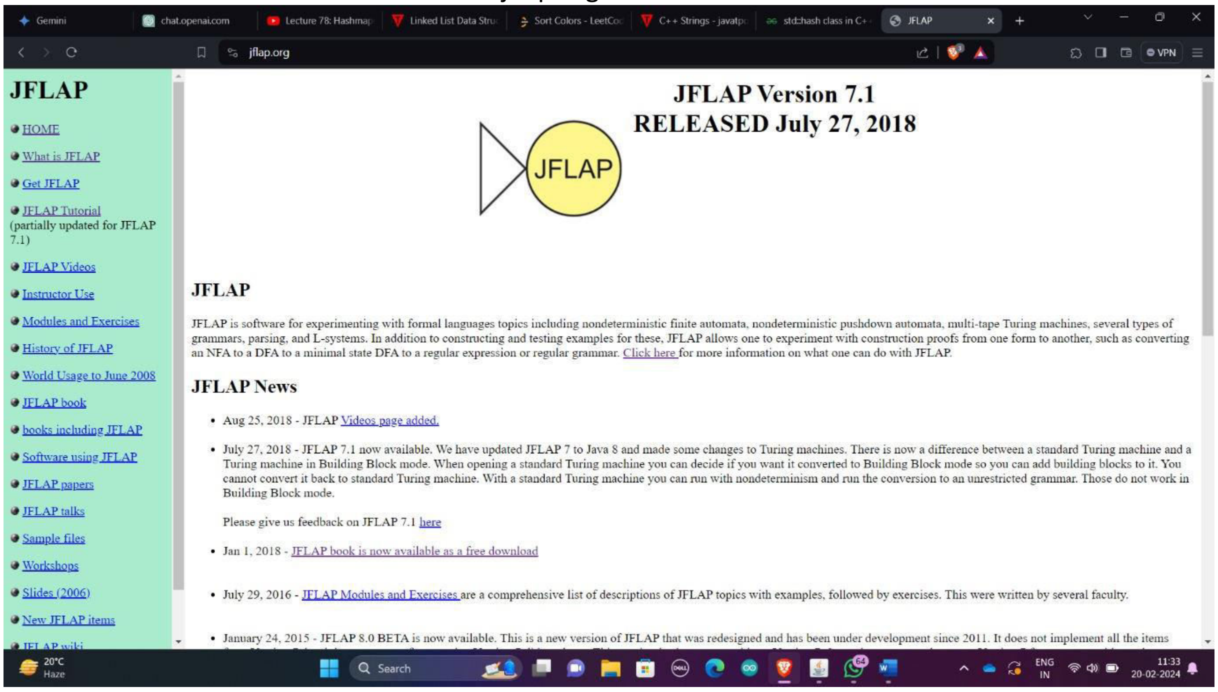
Customization and Extensibility: Users can extend JFLAP's functionality by writing custom automata-related algorithms and plugins using Java.

Educational Tool: JFLAP is widely used in academic settings as a teaching aid for courses in formal languages, automata theory, computability, and complexity theory. Its intuitive interface and interactive features make it an effective tool for illustrating abstract concepts.

Overall, JFLAP serves as a versatile tool for both learning and teaching theoretical computer science concepts related to formal languages and automata, providing a hands-on approach to exploring abstract computational models.

**Steps to download:**

1. Visit the JFLAP website at jflap.org.



JFLAP

HOME
What is JFLAP
Get JFLAP
JFLAP Tutorial (partially updated for JFLAP 7.1)
JFLAP Videos
Instructor Use
Modules and Exercises
History of JFLAP
World Usage to June 2008
JFLAP book
books including JFLAP
Software using JFLAP
JFLAP papers
JFLAP talks
Sample files
Workshops
Slides (2006)
New JFLAP items

JFLAP Version 7.1
RELEASED July 27, 2018

JFLAP

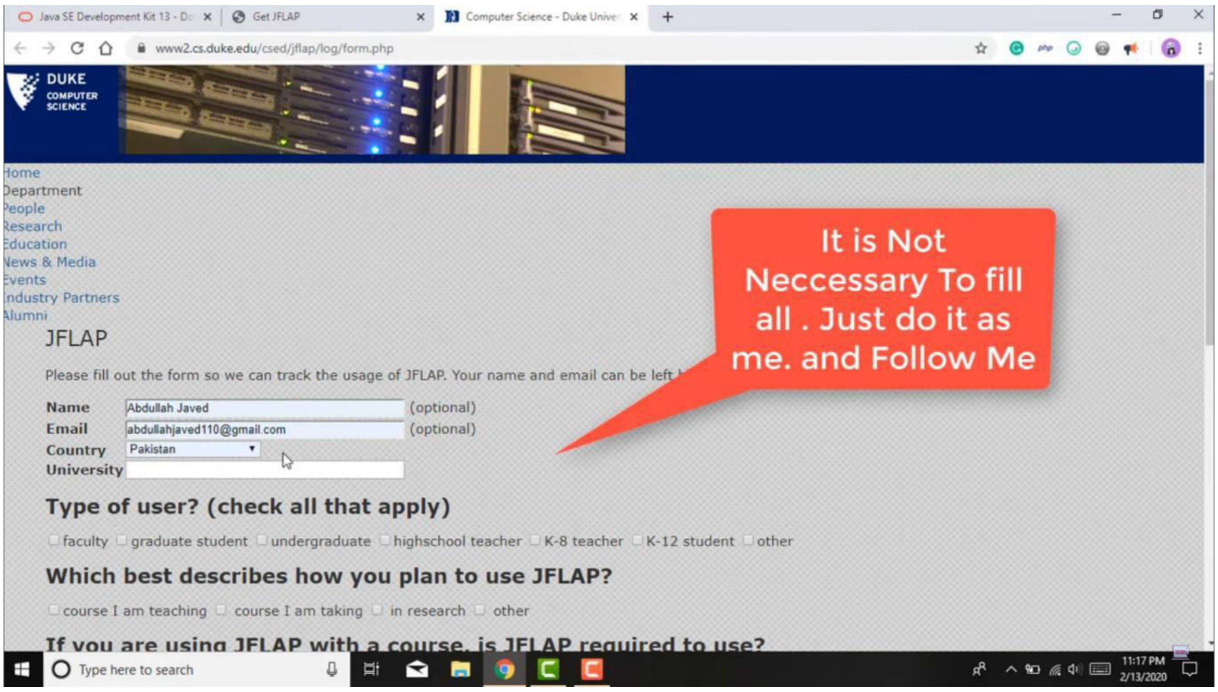
JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, parsing, and L-systems. In addition to constructing and testing examples for these, JFLAP allows one to experiment with construction proofs from one form to another, such as converting an NFA to a DFA to a minimal state DFA to a regular expression or regular grammar. Click here for more information on what one can do with JFLAP.

JFLAP News

- Aug 25, 2018 - JFLAP Videos page added.
- July 27, 2018 - JFLAP 7.1 now available. We have updated JFLAP 7 to Java 8 and made some changes to Turing machines. There is now a difference between a standard Turing machine and a Turing machine in Building Block mode. When opening a standard Turing machine you can decide if you want it converted to Building Block mode so you can add building blocks to it. You cannot convert it back to standard Turing machine. With a standard Turing machine you can run with nondeterminism and run the conversion to an unrestricted grammar. Those do not work in Building Block mode.

  Please give us feedback on JFLAP 7.1 here
- Jan 1, 2018 - JFLAP book is now available as a free download
- July 29, 2016 - JFLAP Modules and Exercises are a comprehensive list of descriptions of JFLAP topics with examples, followed by exercises. This were written by several faculty.
- January 24, 2015 - JFLAP 8.0 BETA is now available. This is a new version of JFLAP that was redesigned and has been under development since 2011. It does not implement all the items

2. Navigate to the "JFLAP SOFTWARE" section or directly access the download page at jflap.org/jflaptmp.



It is Not Neccessary To fill all . Just do it as me. and Follow Me

www2.cs.duke.edu/csed/jflap/log/form.php

Home
Department
People
Research
Education
News & Media
Events
Industry Partners
Alumni

JFLAP

Please fill out the form so we can track the usage of JFLAP. Your name and email can be left

| | | |
|---|---|---|
| Name | Abdullah Javed | (optional) |
| Email | abdullahjaved110@gmail.com | (optional) |
| Country | Pakistan | |
| University | | |

Type of user? (check all that apply)

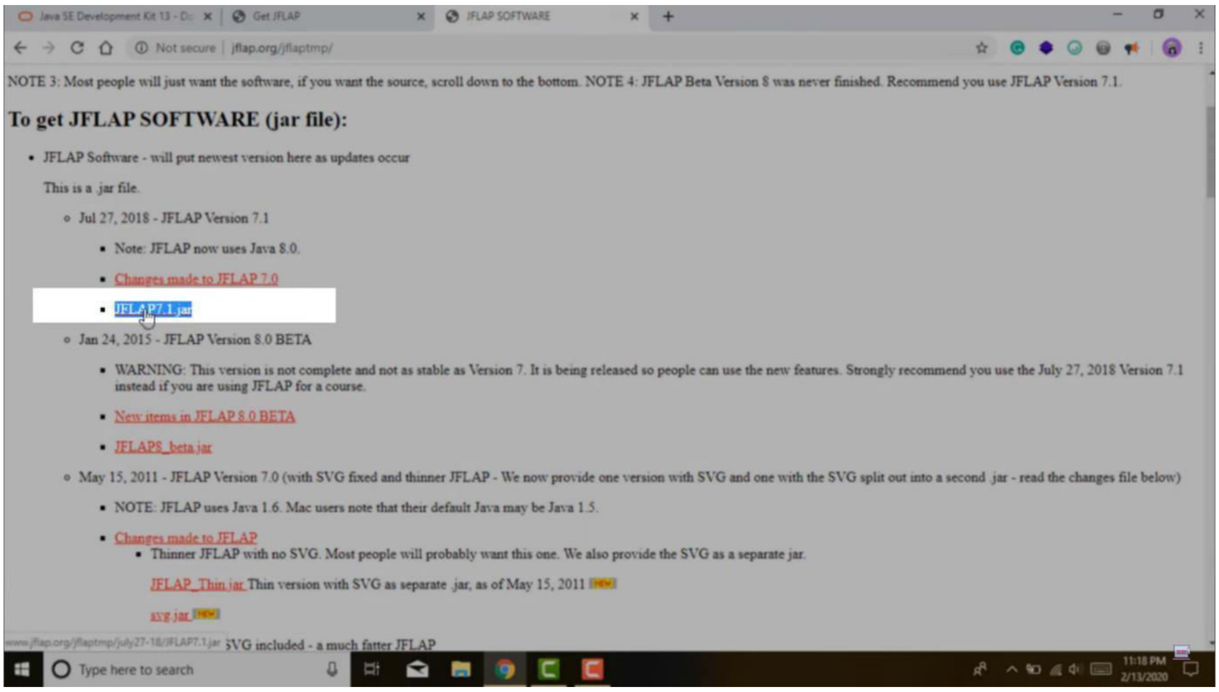
☐ faculty ☐ graduate student ☐ undergraduate ☐ highschool teacher ☐ K-8 teacher ☐ K-12 student ☐ other

Which best describes how you plan to use JFLAP?

☐ course I am teaching ☐ course I am taking ☐ in research ☐ other

If you are using JFLAP with a course, is JFLAP required to use?

3. Look for the latest version, currently recommended as JFLAP Version 7.1.



NOTE 3: Most people will just want the software, if you want the source, scroll down to the bottom. NOTE 4: JFLAP Beta Version 8 was never finished. Recommend you use JFLAP Version 7.1.

**To get JFLAP SOFTWARE (jar file):**

- JFLAP Software - will put newest version here as updates occur

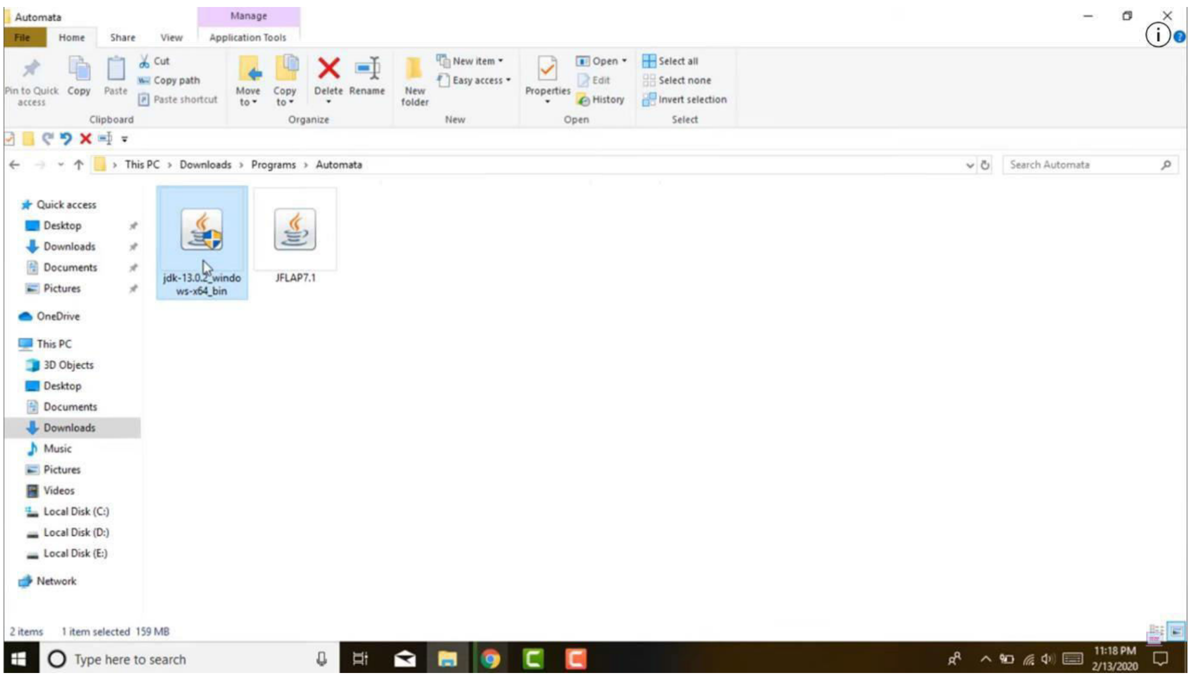
  This is a .jar file.
  - Jul 27, 2018 - JFLAP Version 7.1
    - Note: JFLAP now uses Java 8.0.
    - Changes made to JFLAP 7.0
    - JFLAP7.1.jar
  - Jan 24, 2015 - JFLAP Version 8.0 BETA
    - WARNING: This version is not complete and not as stable as Version 7. It is being released so people can use the new features. Strongly recommend you use the July 27, 2018 Version 7.1 instead if you are using JFLAP for a course.
    - New items in JFLAP 8.0 BETA
    - JFLAP8_beta.jar
  - May 15, 2011 - JFLAP Version 7.0 (with SVG fixed and thinner JFLAP - We now provide one version with SVG and one with the SVG split out into a second .jar - read the changes file below)
    - NOTE: JFLAP uses Java 1.6. Mac users note that their default Java may be Java 1.5.
    - Changes made to JFLAP
      - Thinner JFLAP with no SVG. Most people will probably want this one. We also provide the SVG as a separate jar.

      JFLAP_Thin.jar Thin version with SVG as separate .jar, as of May 15, 2011 NEW

      svg.jar NEW

www.jflap.org/jflaptmp/july27-18/JFLAP7.1.jar SVG included - a much fatter JFLAP

4. Click on the provided link to download the JFLAP software (JAR file).


Automata
This PC › Downloads › Programs › Automata
jdk-13.0.2_windows-x64_bin
JFLAP7.1
2 items 1 item selected 159 MB

5. Once downloaded, you can run JFLAP by double-clicking the JFLAP.jar file on Windows. If that doesn't work, ensure you have Java 1.4 or later installed.

6. To run JFLAP, follow these steps:

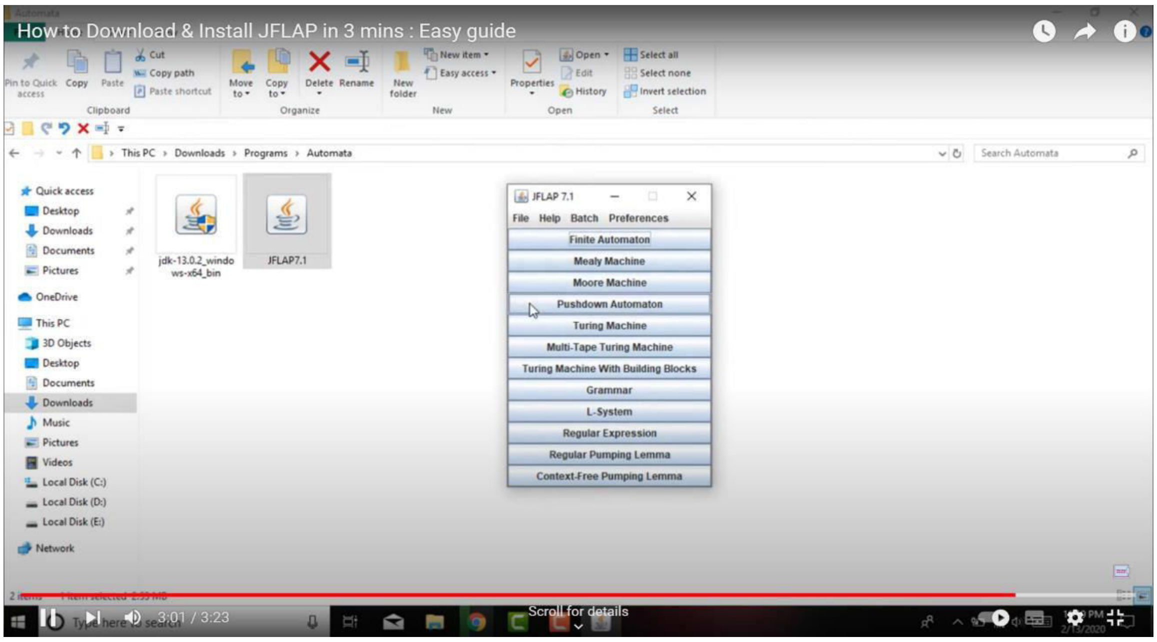
7. Double-click Method (Windows):

8. Double click on the JFLAP.jar file. If this method doesn't work, ensure Java 1.4 or later is installed

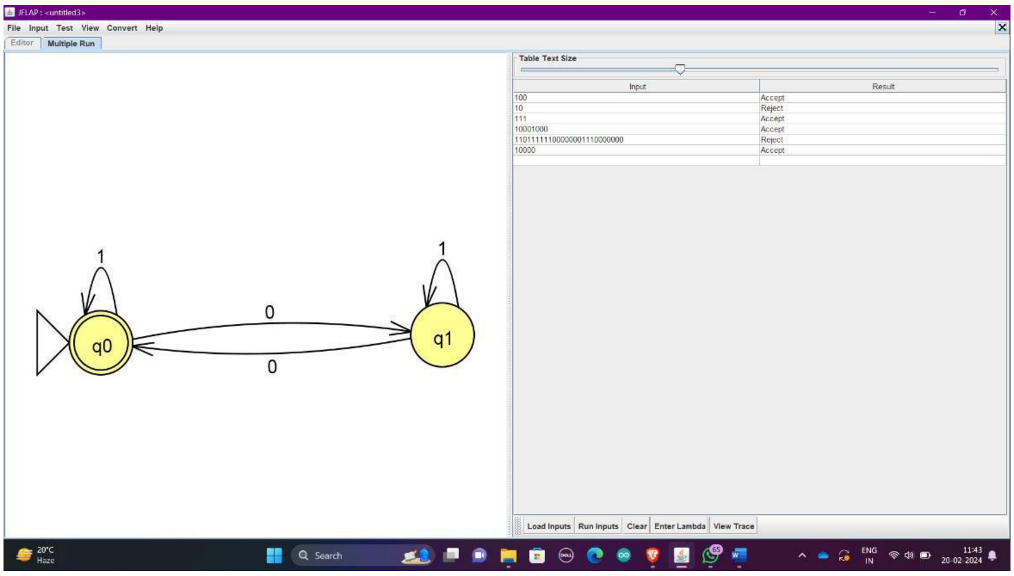
9. Command Line Method:

10. Navigate to the directory containing the JFLAP7.0.jar file.
11. Open a command/console window.
12. Execute the command: java -jar JFLAP7.0.jar [3].
13. Alternative Double-Click Method:

14. Ensure you have Java 6 (Java 1.6) or later installed.
15. Download the JFLAP.jar file from the provided link.
16. Double-click the downloaded JFLAP.jar file to run JFLAP



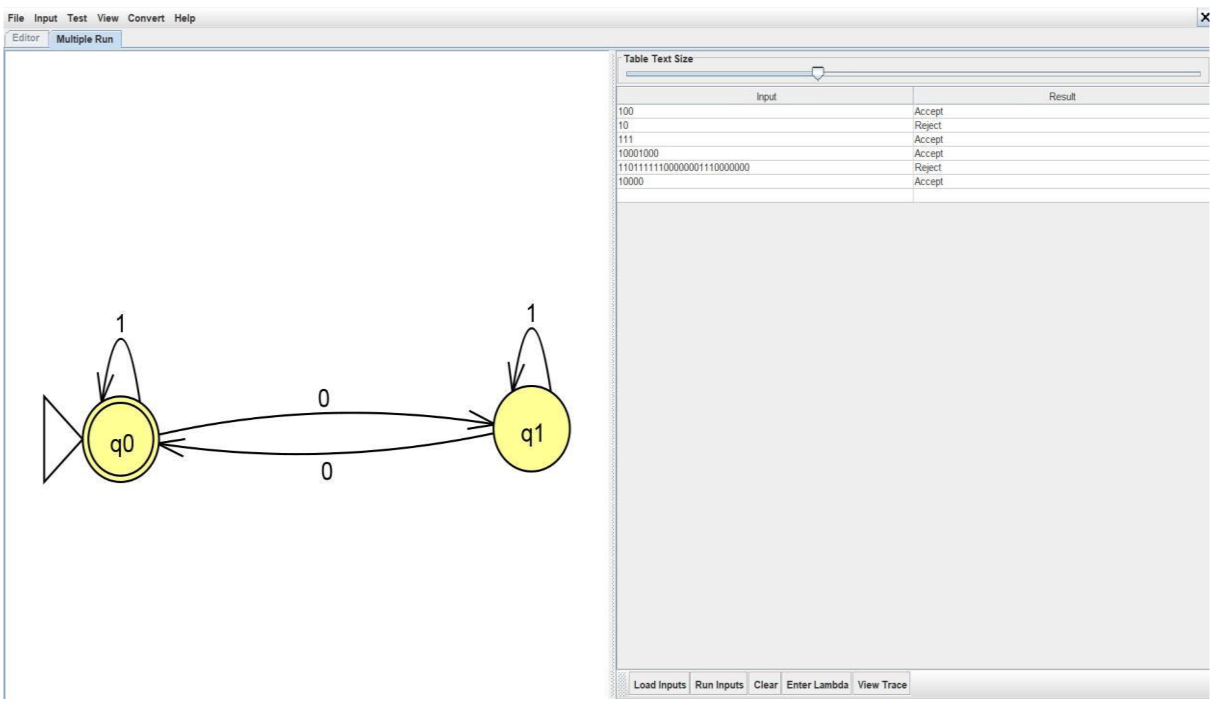
How to Download & Install JFLAP in 3 mins : Easy guide
This PC › Downloads › Programs › Automata
jdk-13.0.2_windows-x64_bin
JFLAP7.1
JFLAP 7.1
File Help Batch Preferences
Finite Automaton
Mealy Machine
Moore Machine
Pushdown Automaton
Turing Machine
Multi-Tape Turing Machine
Turing Machine With Building Blocks
Grammar
L-System
Regular Expression
Regular Pumping Lemma
Context-Free Pumping Lemma
3:01 / 3:23



| Input | Result |
|---|---|
| 100 | Accept |
| 10 | Reject |
| 111 | Accept |
| 10001000 | Accept |
| 110111111000000001110000000 | Reject |
| 10000 | Accept |

# EXPERIMENT-2

Q1) Design a DFA which will accept all the strings containing even number of 0's over an alphabet (0, 1) and write a program to implement the DFA.



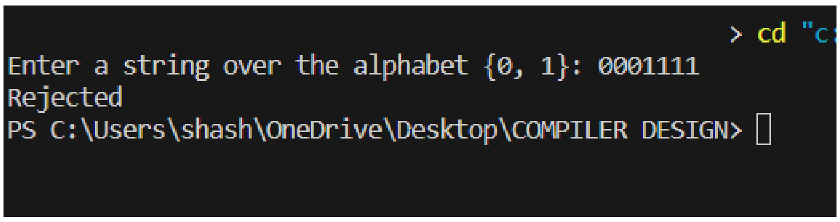
| Input | Result |
|---|---|
| 100 | Accept |
| 10 | Reject |
| 111 | Accept |
| 10001000 | Accept |
| 11011111100000001110000000 | Reject |
| 10000 | Accept |

## Code –

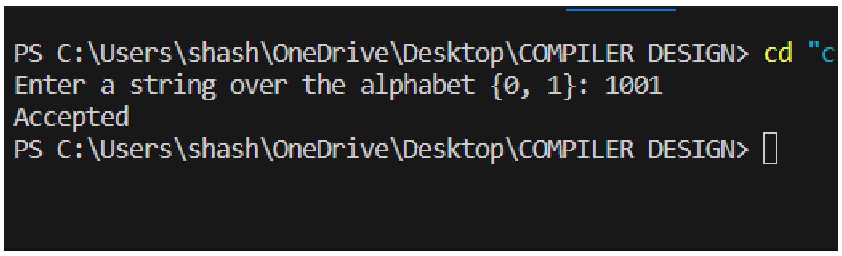
```
#include <iostream>
#include <string>

using namespace std;

// Function to simulate the DFA
bool isEvenNumberOfZeros(string input) {
   int currentState = 0; // Initial state q0
   for (char c : input) {
      if (c == '0') {
         currentState = (currentState == 0) ? 1 : 0; // Transition based on current state and input
      }
   }
   return currentState == 0 && input.find('0') != string::npos; // Reject if the final state is q0 and '0' is
present in the input
}
```

```
int main() {
    string input;
    cout << "Enter a string over the alphabet {0, 1}: ";
    cin >> input;

    if (isEvenNumberOfZeros(input)) {
        cout << "Accepted" << endl;
    } else {
        cout << "Rejected" << endl;
    }

    return 0;
}
```

## OUTPUT –



```
                                                    > cd "c:
Enter a string over the alphabet {0, 1}: 0001111
Rejected
PS C:\Users\shash\OneDrive\Desktop\COMPILER DESIGN> 
```



```
PS C:\Users\shash\OneDrive\Desktop\COMPILER DESIGN> cd "c
Enter a string over the alphabet {0, 1}: 1001
Accepted
PS C:\Users\shash\OneDrive\Desktop\COMPILER DESIGN> 
```