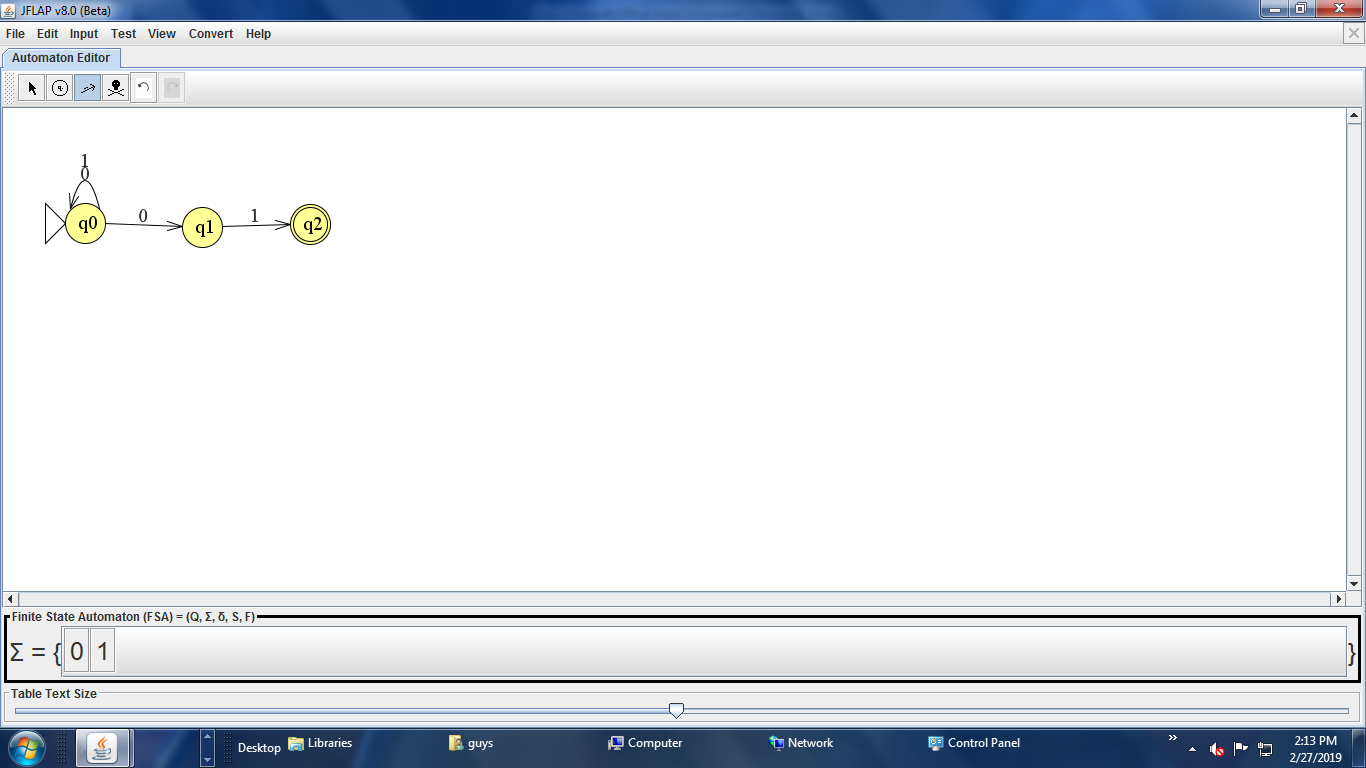
**EXP:1**

**CONVERSION FROM NFA TO DFA**

**AIM:**

Convert the following NFA to DFA



**ALGORITHM:**

* Start the JFLAP file.
* Add required states.
* Add required symbols of transition states
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP file.
* Click Finite Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**Transition table for the given question**

|  |  |  |
| --- | --- | --- |
|  | Input symbols |  |
| States | 0 | 1 |
| Q0 | {Q1,Q0} | {QO} |
| Q1 | {ⱷ} | {Q2} |
| \*Q2 | {ⱷ} | {ⱷ} |

**SUBSET CONSTRUCTION METHOD**

To make a transition from the new state {A}

Over an alphabet ∑={0,1}

δ (A,0)= δ({Q0},O)={Q0,Q1} B(Intermediate State)

δ(A,1)= δ({Q0,1})={Q0} A(intial state)

δ(B,0)= δ({Q0,Q1,0})={Q0,Q1} B(Intermediate State)

δ(B,1)= δ({Q0,Q1}.1)={Q0,Q2} C(Final State)

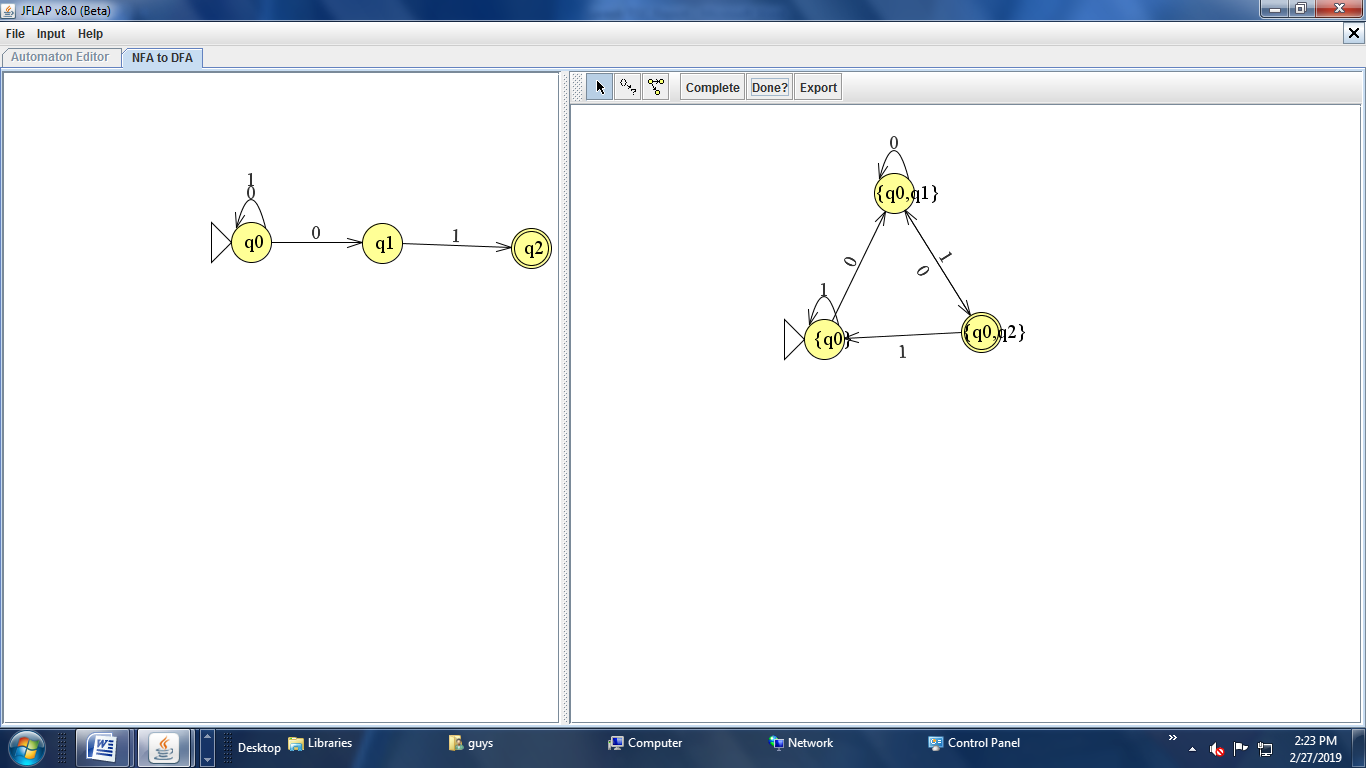
δ(C,0)= δ({Q0,Q2},0)={Q0,Q1} B(Intermediate State)

δ(C,1)= δ ({Q0,Q2},1)={Q0} A(Intial State)

**TRANSTITION TABLE:**

|  |  |  |
| --- | --- | --- |
| STATES | INPUT SYMBOLS | |
| 0 | 1 |
| A[Q0]  B[Q0,1]  \*C[Q0,Q2] | B[Q0,Q1]  B[Q0,Q1]  B[Q0,Q1] | A[Q0]  C[Q0]  A[Q0] |

**SCREENSHOTS:**

****

**RESULT:**

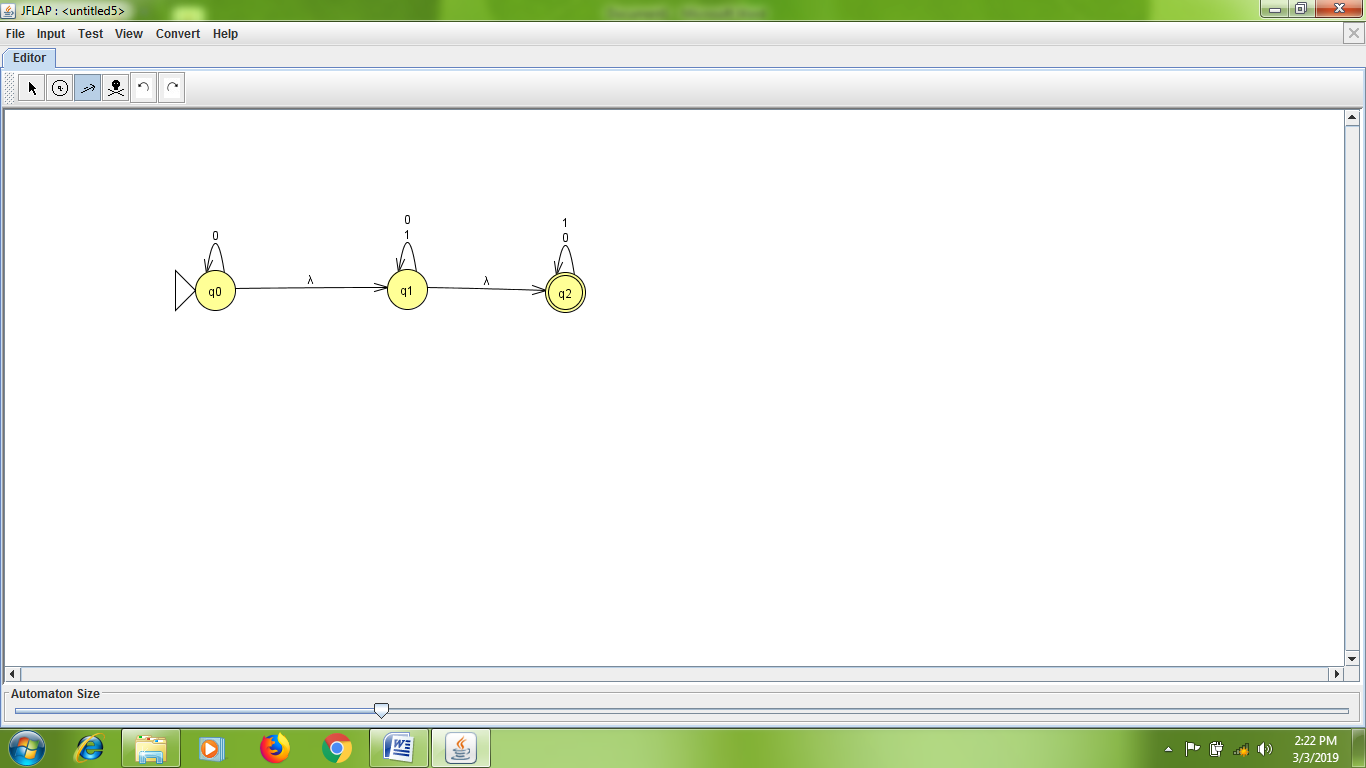
Thus the given NFA is successfully converted to DFA

**EXP:2**

**CONVERT FROM €-NFA TO NFA**

**AIM:**

To design a NFA for the following €-NFA.



**ALGORITHM:**

* Start the JFLAP file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP file.
* Click Finite Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**SOLUTION:**Step1: To find €-closure of all the states

€-closure(q0)={ q0 q1,q2 }→A

€-closure(q1)= { q1,q2}→B

€-closure(q2)={q2}→C

Step 2: To make a transition over an alphabet∑={0,1} for the newstate A,B,C

ᵟ(A,0) =€-closure(ᵟ({q0,q1,q2},0)) =€-closure {q0,q1,q2} = { q0,q1,q2}→A(final state)

ᵟ(A,1) =€-closure(ᵟ({q0,q1,q2},1)) =€-closure { q1,q2} = {q1,q2}→B(final state)

ᵟ(B,0) =€-closure(ᵟ({q1,q2},0)) =€-closure {q1,q2} = { q1,q2}→B(final state)

ᵟ(B,1)=€-closure(ᵟ({q1,q2},1)) =€-closure {q1,q2} = {q1,q2}→B(final state)

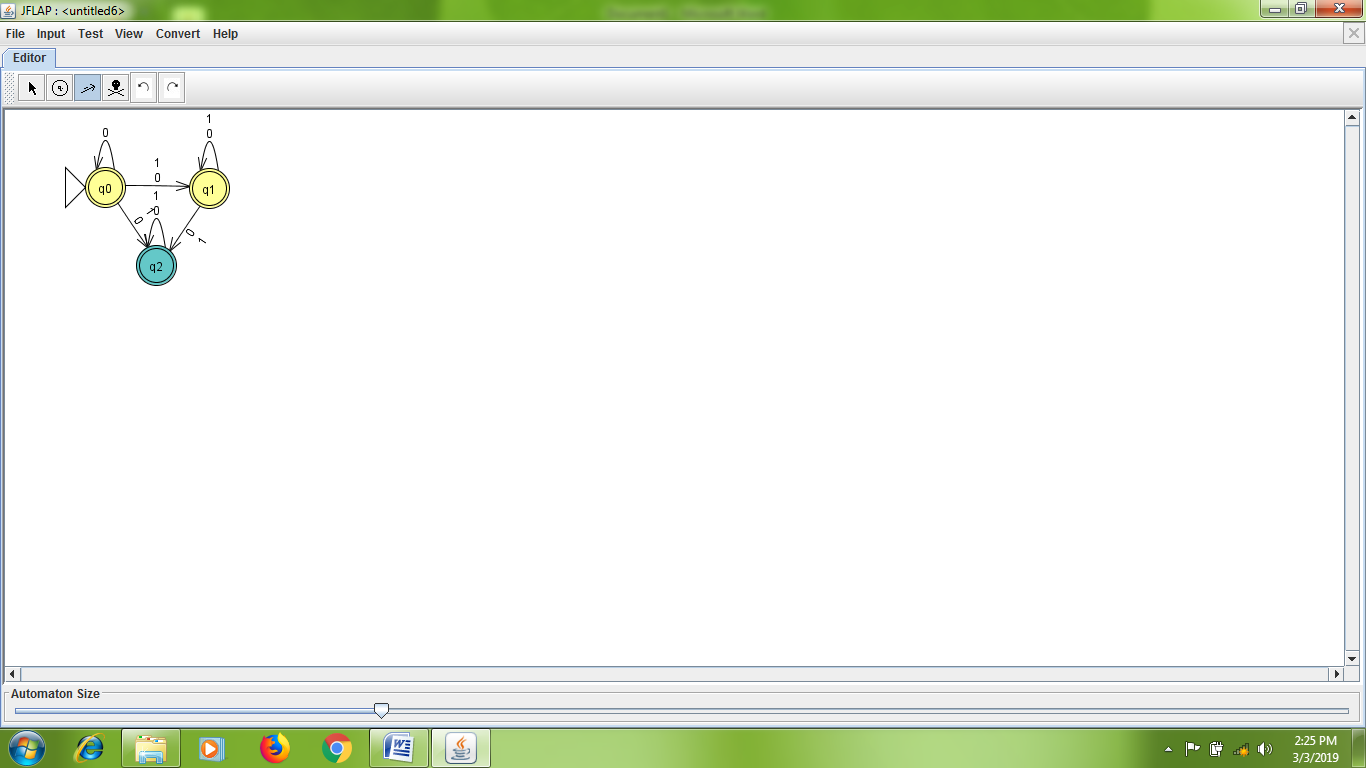
ᵟ(C,0) =€-closure(ᵟ({q2},0)) =€-closure { q2} = {q2}→C(final state)

ᵟ(C,1) =€-closure(ᵟ({q2},1)) =€-closure { q2} = {q2}→C(final state)

**TRANSITION TABLE:**

|  |  |  |
| --- | --- | --- |
| States | Inputs | |
| A | B |  |
| →\*A{q0,q1,q2}  \*B{q1,q2}  \*C{q2} | A{q0,q1,q2}  B{q1,q2}  C{q2} | B{q1,q2}  B{q1,q2}  C{q2} |

**SCREENSHOTS:**



**RESULT:**

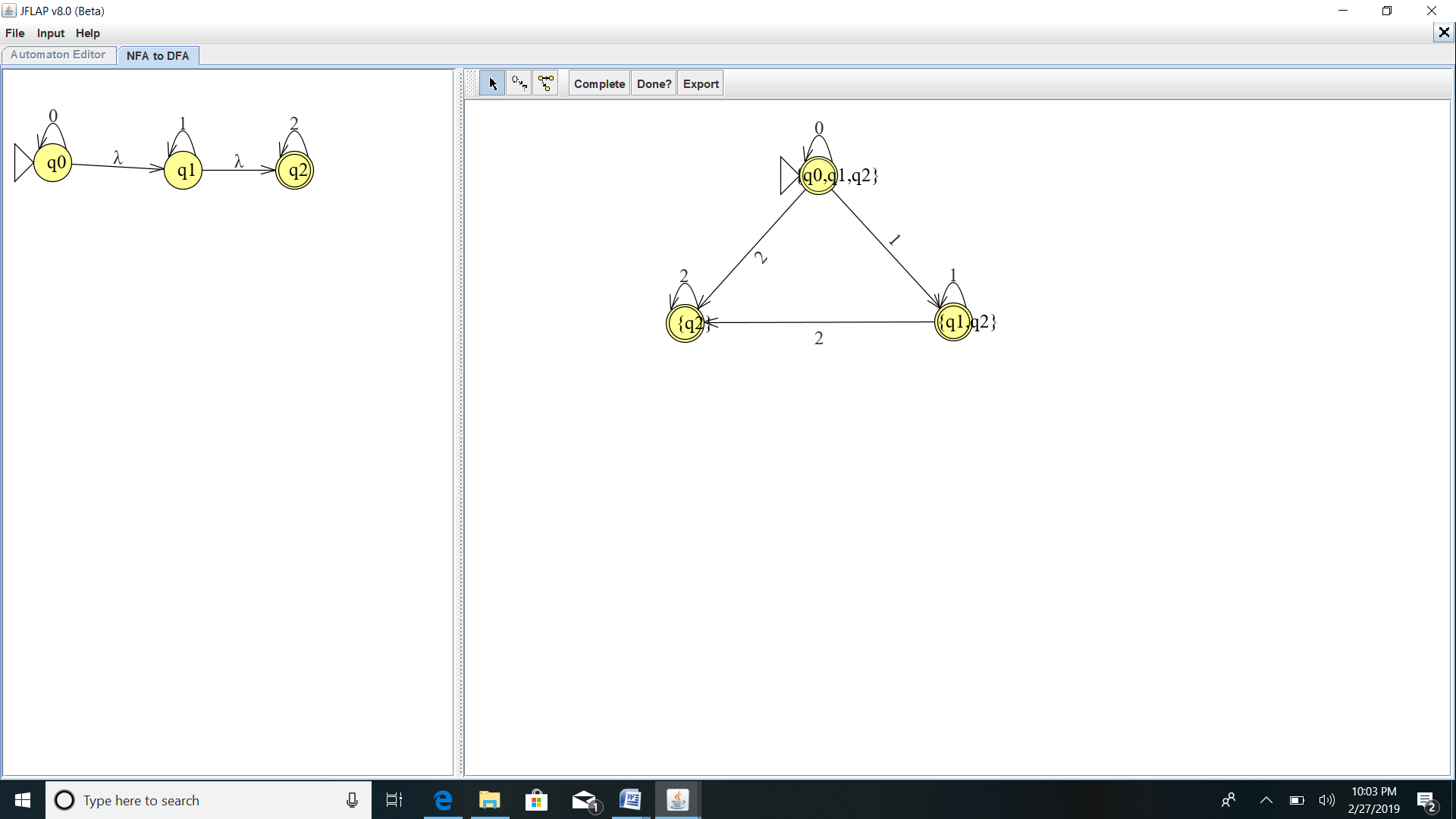
Thus, the given €-NFA has been successfully designed.

**EXP:3**

**CONVERSION FROM ε-NFA TO DFA**

**AIM :**

Convert the following ε-NFA to DFA

****

**ALGORITHM:**

* Start the JFLAP file.
* Add required states.
* Add required symbols of transition states
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP file.
* Click Finite Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

STEP-1:

To find ε-closure of all the states

ε-closure(q0)={q0,q1,q2}

ε-closure(q1)={q1,q2}

ε-closure(q2)={q2}

STEP -2:

**SUBSET CONSTRUCTION METHOD**

The initial state of the given ε-NFA is {q0} So,the initial state of the converting DFA is A {q0,q1,q2} which is ε-closure of (q0)

Then to make a transition from A over an alphabet ∑={0,1,2}

***TRANSITIONS ARE:***

δ (A,0)=ε-closure δ ({q0,q1,q2},0)

= ε-closure(q0)

= {qo,q1,q2} A(Initial and final state)

δ (A,1)=ε-closure δ ({q0,q1,q2},1)

= ε-closure(q1)

={q1,q2} B(final state)

δ (A,2)=ε-closure δ ({q0,q1,q2},2)

= ε-closure(q2)

={q2} C(final state)

δ (B,0)=ε-closure δ ({q1,q2},0)

= ε-closure (φ)

=φ

δ (B,1)=ε-closure δ ({q1,q2},1)

= ε-closure(q1)

={q1,q2} B(final state)

δ (B,2)=ε-closure δ ({q1,q2},2)

= ε-closure(q2)

={q2} C(final state)

δ (C,0)=ε-closure δ ({q2},0)

= ε-closure (φ)

=φ

δ (C,1)=ε-closure δ ({q2},1)

= ε-closure (φ)

=φ

δ (C,2)=ε-closure δ ({q2},2)

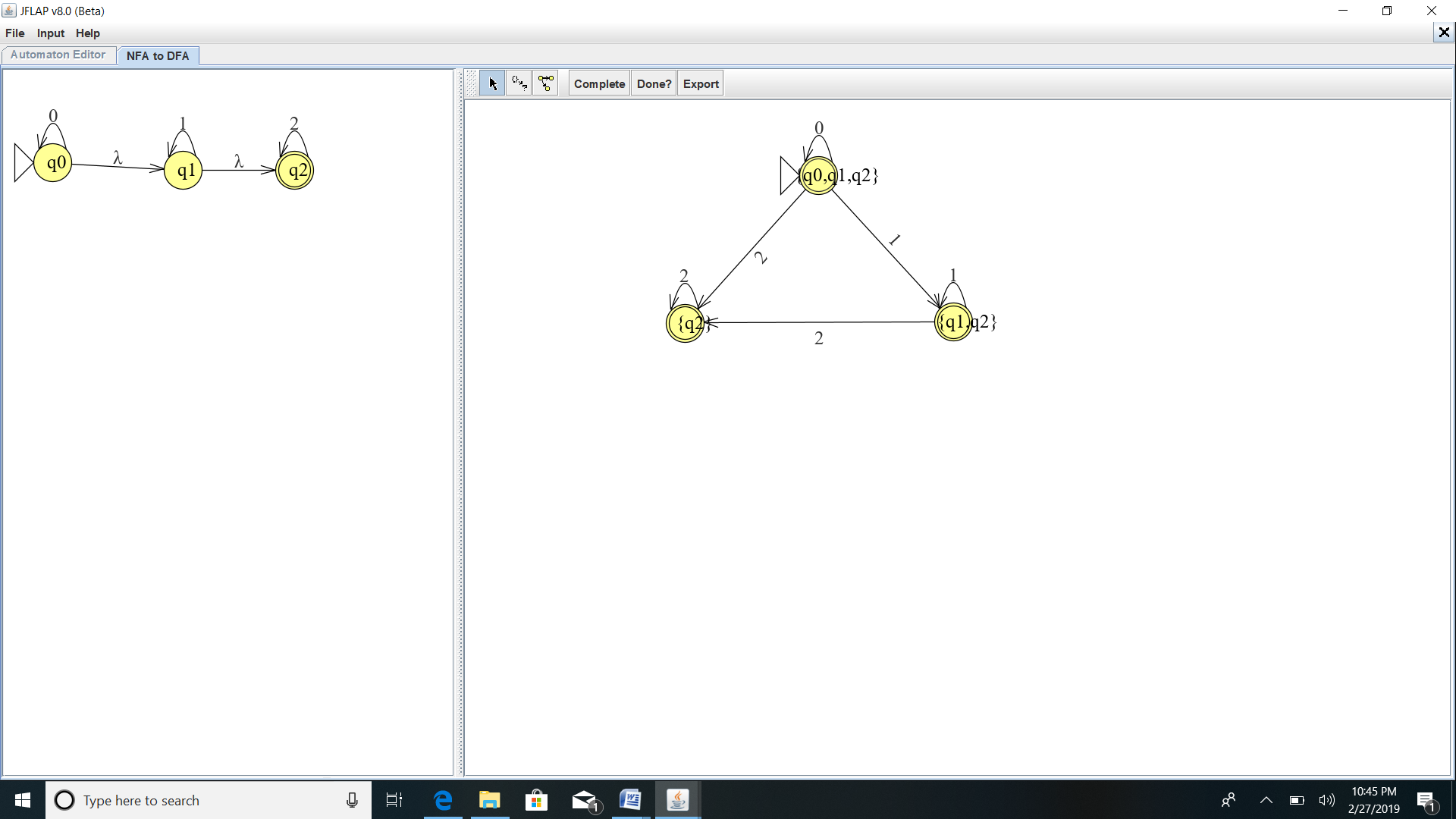
= ε-closure (q2)

={q2} C(final state)

**TRANSITION TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| STATES | INPUT SYMBOLS | | |
| 0 | 1 | 2 |
| \*A{q0,q1,q2}  \*B{q1,q2}  \*C[q2] | A[q0,q1,q2]  Φ  φ | B[q1,q2]  B[q1,q2]  Φ | C[q2]  C[q2]  C[q2] |

**SCREENSHOTS**:



***RESULT:***

Thus the given automata is successfully converted from ε-NFAto DFA

**EXP:4**

**CONVERT FROM RE TO DFA**

**AIM:**

To convert the given RE = (0+1)\* to DFA.

**ALGORITHM:**

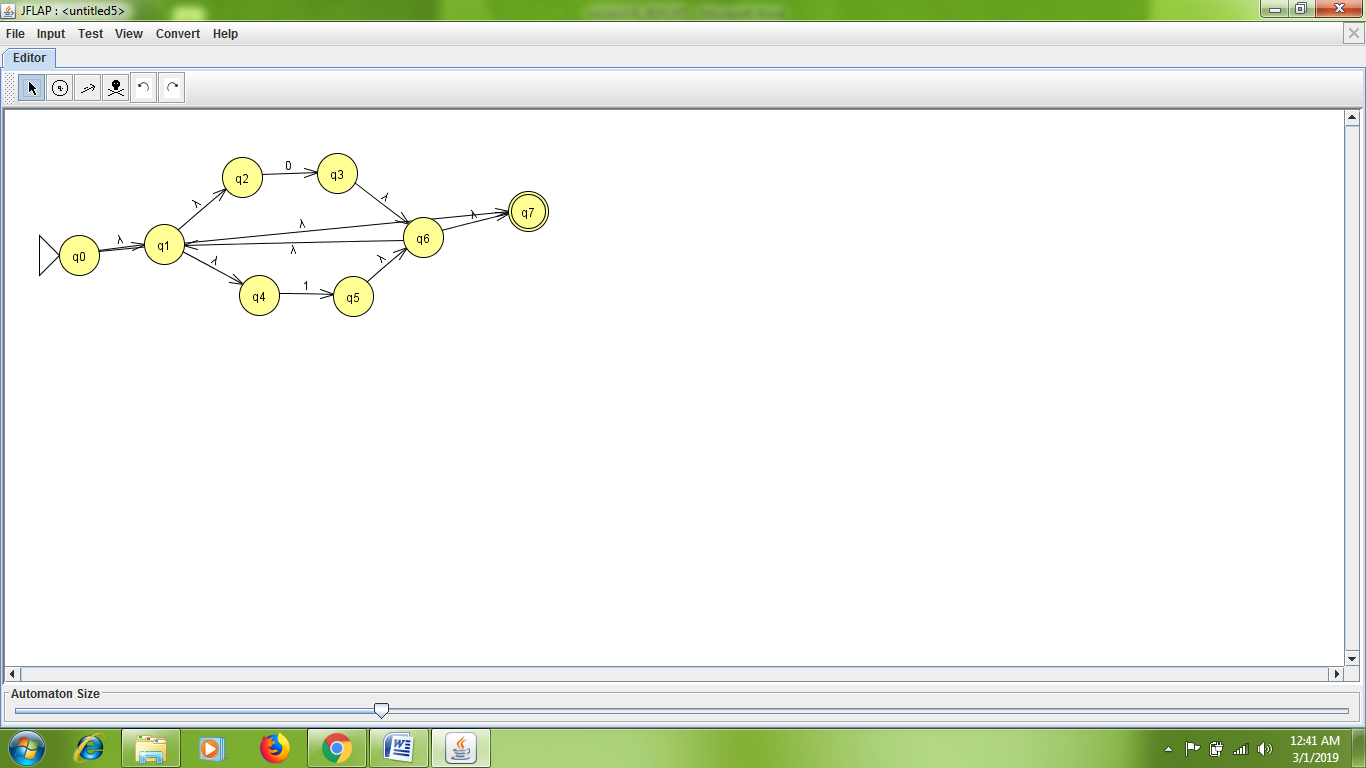
* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Click convert to DFA button.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Finite Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**SOLUTION:**

Step 1: RE to €-NFA by using THOMSONS CONSTRUCTION ALGORITHM



Step 2: €-NFA to DFA by using SUBSET CONSTRUCTION ALGORITHM

€-closure(0)={0,1,2,4,7}

€-closure(1)={2}

€-closure(3)={1,2,3,4,6,7}

€-closure(4)={4}

€-closure(5)={1,2,4,5,6,7}

€-closure(6)={1,2,4,6,7}

€-closure(7)={7}

Transition over an alphabet∑={0,1}

Initial State of €-NFA is 0

Let us assume €-closure(0)={0,1,2,4,7} →A(final state)

ᵟ(A,0) = €-(ᵟ(0,1,2,4,7),0) = €-{3} = {1,2,3,4,6,7} →B(final state)

ᵟ(A,1) = €-(ᵟ(0,1,2,4,7),1) = €-{5} = {1,2,4,5,6,7} →C(final state)

ᵟ(B,0) = €-(ᵟ(0,1,2,3,4,6,7),0) = €-{3} = {1,2,3,4,6,7} →B(final state)

ᵟ(B,1) = €-(ᵟ(0,1,2,3,4,6,7),1) = €-{5} = {1,2,4,5,6,7} →C(final state)

ᵟ(C,0) = €-(ᵟ(0,1,2,4,5,6,7),0) = €-{3} = {1,2,3,4,6,7} →B(final state)

ᵟ(C,1) = €-(ᵟ(0,1,2,4,5,6,7),1) = €-{5} = {1,2,4,5,6,7} →C(final state)

**TRANSITON TABLE:**

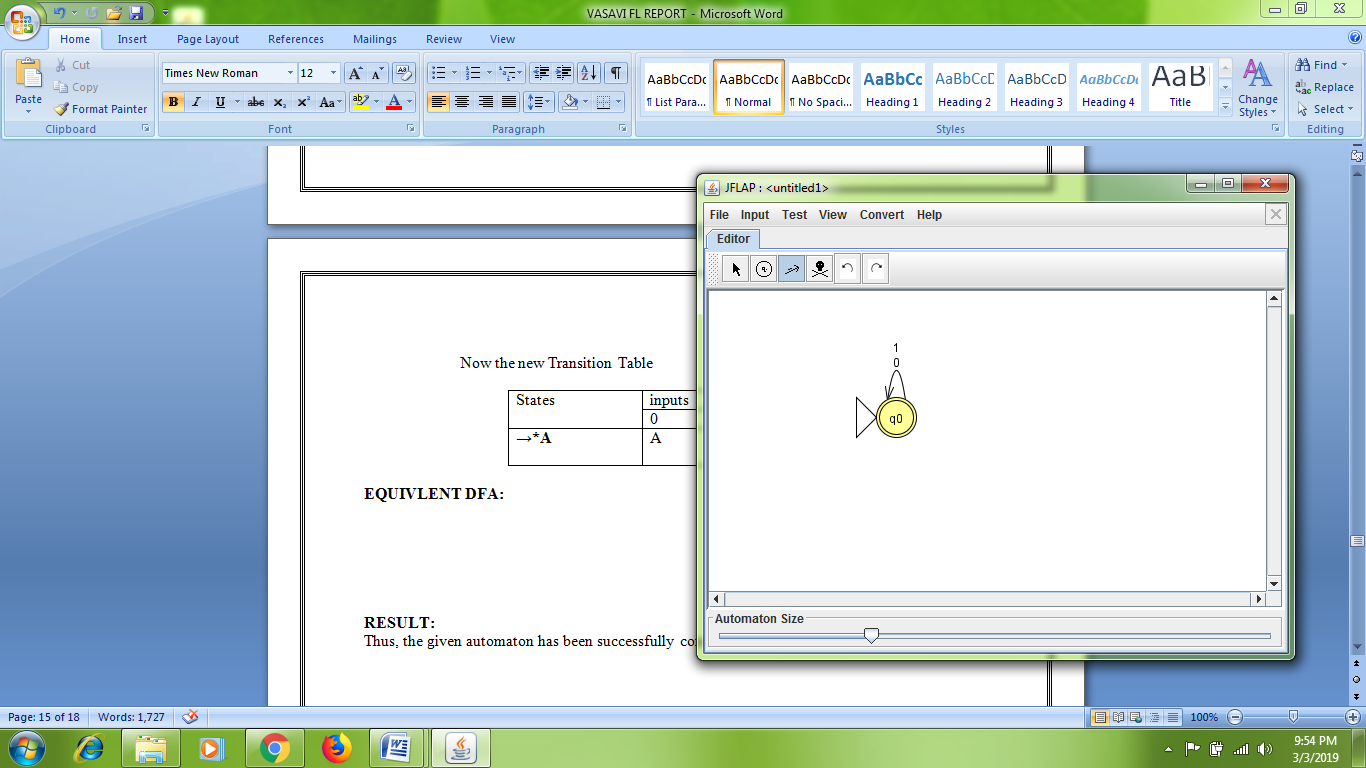
|  |  |  |
| --- | --- | --- |
| states | inputs | |
| **0** | **1** |
| **→\*A**  **\*B**  **\*C** | **B**  **B**  **B** | **C**  **C**  **C** |

Equivalent are A=B=C

Now the new Transition Table

|  |  |  |
| --- | --- | --- |
| States | Inputs | |
| 0 | 1 |
| **→\*A** | A | A |

**EQUIVLENT DFA:**

****

**RESULT:**

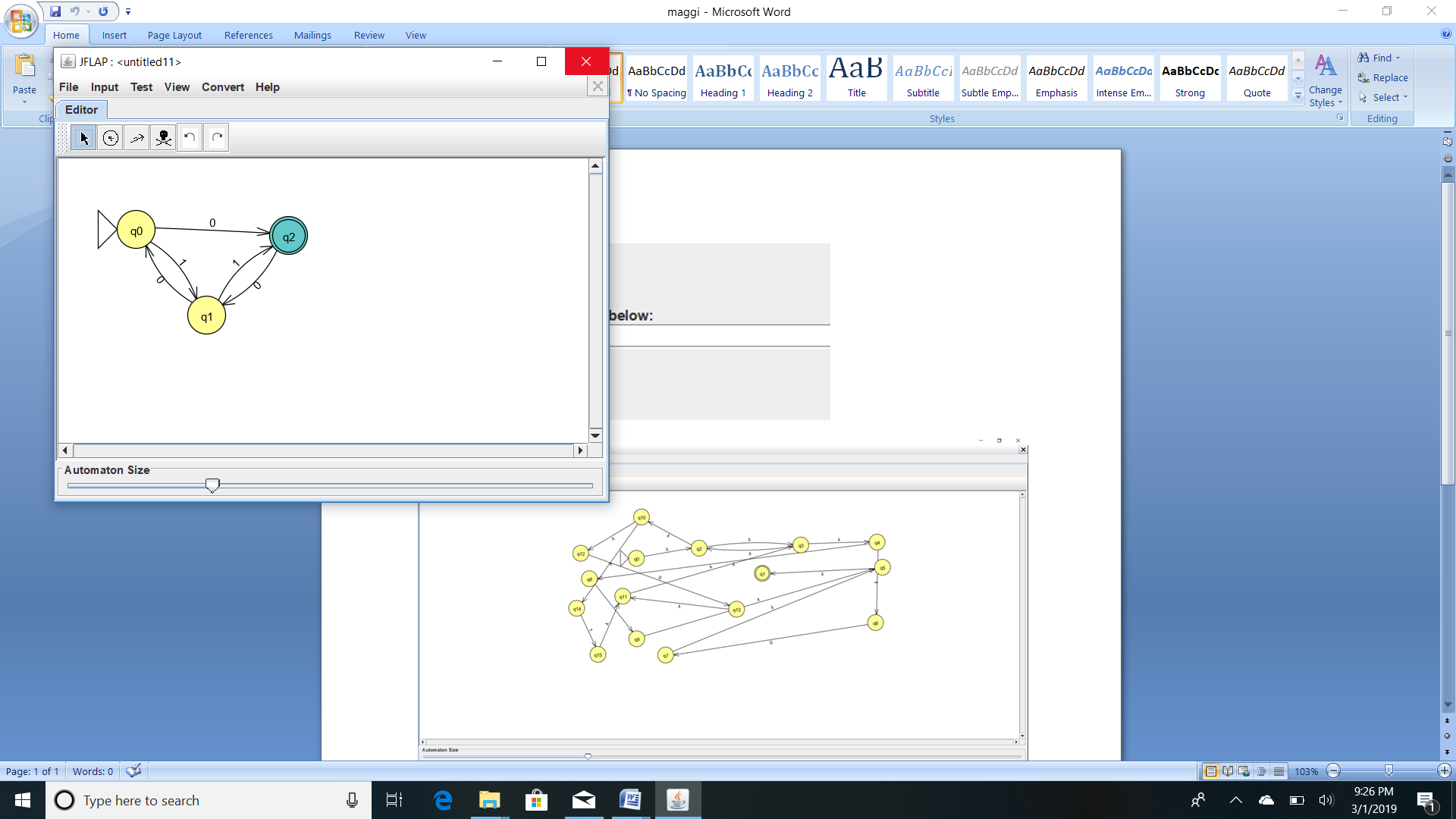
Thus, the given automaton has been successfully converted to DFA.

**EXP:5**

**CONVERSION FROM DFA TO REGULAR EXPRESSION**

**AIM:**

Convert the following automata to DFA



**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Add required symbols of transition states
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Finite Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**SOLUTION:**

STEP:1

The regular expression is denoted by R1j(n)

Where 1→initial state(1)

j→final state(2)

n-number of states(3)

Then we want to find the RE R13(3)

BASIS STEP:

R11(0)  = ε

R12(0)  = 1

R13(0)  = 0

R21(0)  = 0

R22(0)  = ε

R23(0)  =1

R31(0)  = φ

R32(0)  = 0

R33(0)  = ε

INDUCTIVE STEP:

Let k=k+1

The formula is

|  |
| --- |
| Rij(k) **=** Rij(k-1)  **+** Rik(k-1)(Rkk(k-1) **)\***Rkj(k-1) |

Let k=1

R11(1)  = R11(0)  + R11(0)  ( R11(0)  ) \* R11(0)

= ε+ ε. ε\*.ε

= ε

R12(1)  = R12(0)  + R11(0)  ( R11(0)  ) \* R12(0)

=1+ ε. ε\*1

=1+1

=1

R13(1)  = R13(0)  + R11(0)  ( R11(0)  ) \* R13(0)

=0+ε.ε\*0

=0+0

=0

R21(1)  = R21(0)  + R21(0)  ( R11(0)  ) \* R11(0)

=0+0ε\*ε

=0+0

=0

R22(1)  = R22(0)  + R21(0)  ( R11(0)  ) \* R12(0)

=ε+0 ε\* ε

=0+0

=0

R23(1)  = R23(0)  + R21(0)  ( R11(0)  ) \* R13(0)

=1+0 ε\*0

=1+0

R31(1)  = R31(0)  + R31(0)  ( R11(0)  ) \* R11(0)

= φ+φ ε\*0

=φ

R32(1)  = R32(0)  + R31(0)  ( R11(0)  ) \* R12(0)

=0+φ

=0

R33(1)  = R33(0)  + R31(0)  ( R11(0)  ) \* R13(0)

=ε+φ

` =ε

Now let k=2

R11(2)  = R11(1)  + R12(1)  ( R22(1)  ) \* R21(1)

=ε+1 .(01)\*.0

=(01)\*

R12(2)  = R12(1)  + R12(1)  ( R22(1)  ) \* R22(1)

=1+1(01)\*(01)

=1+(01)\*1

=1.(01)\*

R13(2)  = R12(1)  + R12(1)  ( R22(1)  ) \* R23(1)

=0+1(01)\*.(1+0)

=0+1(0.1)\*(0+1)

R21(2)  = R21(1)  + R22(1)  ( R22(1)  ) \* R12(1)

=0+01(01)\*0

=0.(01)\*

R22(2)  = R22(1)  + R22(1)  ( R22(1)  ) \* R22(1)

=01+(01)(01)\*(01)

=01+(01)\*

=(01)\*

R23(2)  = R23(1)  + R22(1)  ( R22(1)  ) \* R23(1)

=(0+1)+(01)(0.1)\*(1+0)

=(0+1)(01)\*

R31(2)  = R31(1)  + R32(1)  ( R22(1)  ) \* R21(1)

=φ+0(01)\*.0

=0.(01)\*

R32(2)  = R32(1)  + R32(1)  ( R22(1)  ) \* R22(1)

=0+0(01)\*(0)

=(01)\*0

R33(2)  = R33(1)  + R32(1)  ( R22(1)  ) \* R22(1)

=ε+0(01)\*(0)

=(01)\*0

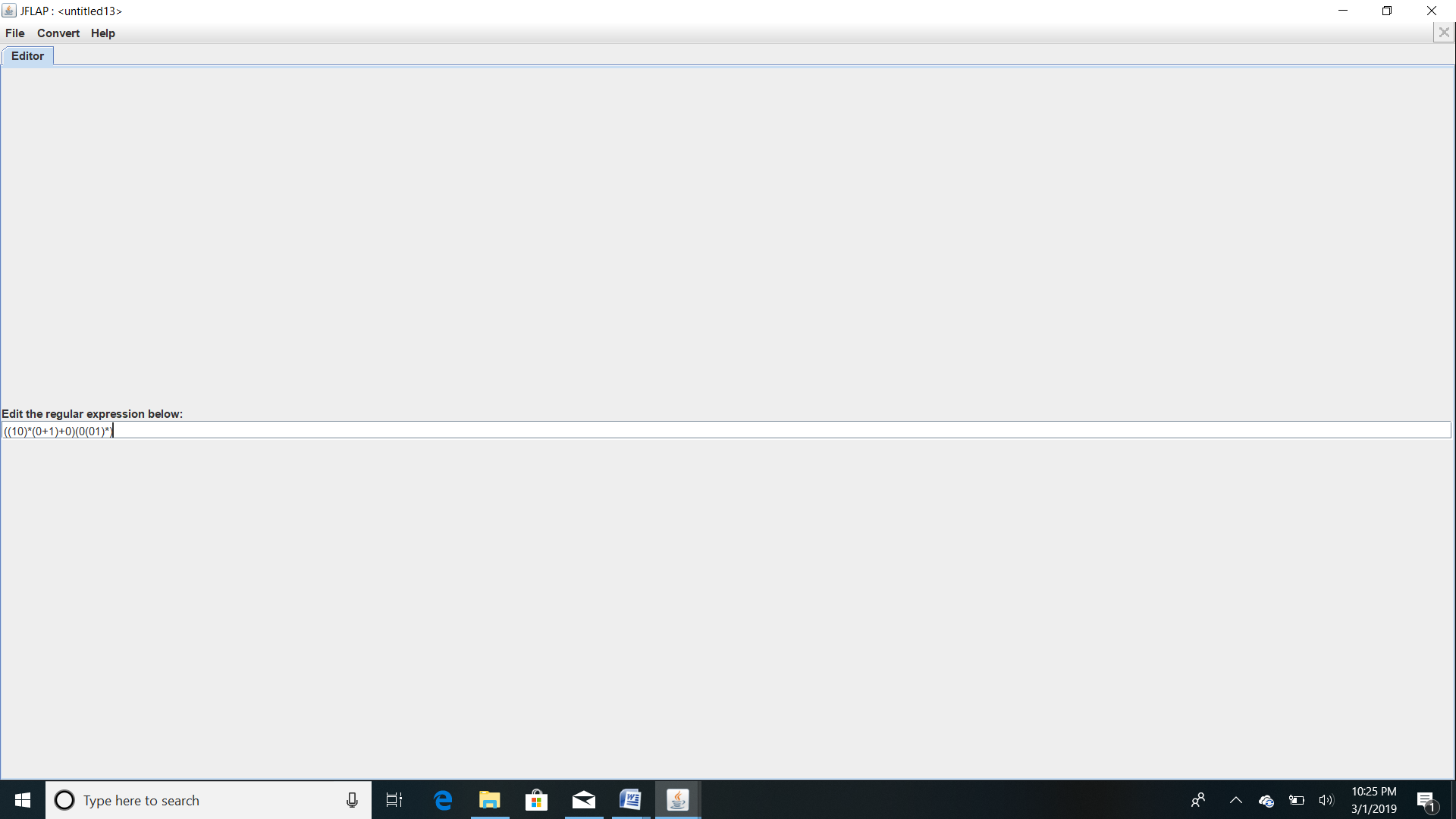
The required one is

R13(3)  = R13(2)  + R13(2)  ( R33(2)  ) \* R33(2)

= [0+1(0.1)\*(0+1)]+ [0+1(0.1)\*(0+1)((01)\*0))((01)\*0)]

== [0+1(0.1)\*(0+1)][ε+((01)\*0)]

***SCREENSHOTS:***



***RESULT:***

Thus the given automata is successfully converted from DFA TO RE

**EXP:6**

**PUMPING LEMMA**

**AIM:**To check wheather the given language satisfies Pumping lemma conditions or not.

**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Pumping lemma button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**THEOREM:**

let “L” be a regular language then exits a constant n which depends on the regular

language “L”. such that for every string w. |w|>=n then we can break w into the strings that is

w=xyz. such that

1. Y ≠ €
2. |xy| < n
3. |Y| >0
4. For all k>=0 then String xykz Є L

**SOLUTION:**

The given language is L={anbn/n>=0}

Assume “L” be an regular language

Let n=2

L={a2b2}=>w=aabbi.e, |w|=4

The pumping lemma conditions are

(i)y≠€ (iii)|xy|<n

(ii)|y|>0 (iv)For all k≥0 then the string xykЄ L

The string is W=aabb

Let X=a y=ab z=b

i)y≠€ y=aby≠€ satisfied

ii)|y|>0 y=ab |y|=2 |y|>0 satisfied

iii)|xy|<n x=a y=abxy=aab |xy|=3

iv)For all k>0 the string xykz

let k=2

xy2z=a(ab)2b=aababb€2

Not satisfied

iii)|xy|≤n x=aa y=a xy=aaa

|xy|=3 not satisfied

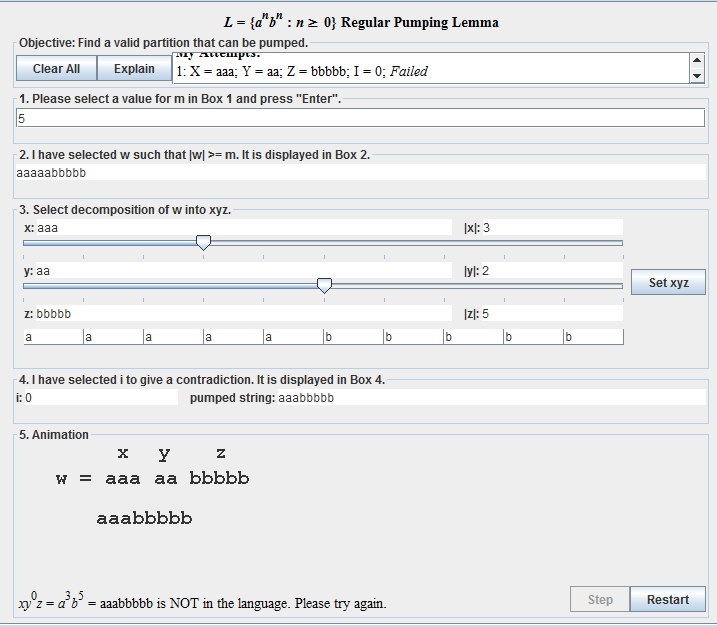
iv) For all k≥0 then string xykzЄ z

k=1

xykz=aa(a)1bbb =aaabbb satisfied

xykz Є L

**SCREENSHOTS:**



**RESULT:** Thus, the conditions successfully checked.

**Exp no:7**

**PUSH-DOWNAUTOMATA[PDA]**

**AIM:**

To construct a pushdown automata.

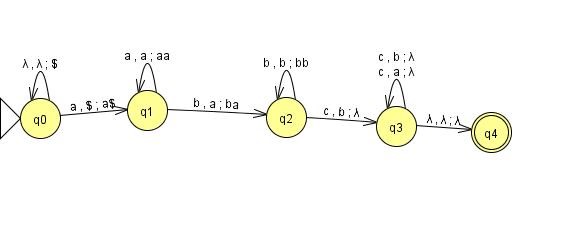
**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Pushdown Automaton button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**SCREENSHOTS:**

PDA that accepts L = {an bn cn  n ≥ 1

**RESULT:**

Thus, the given PDA is constructed successfully

**Exp no:8**

**CFG from PDA**

**AIM:**

To construct a context free grammer.

**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

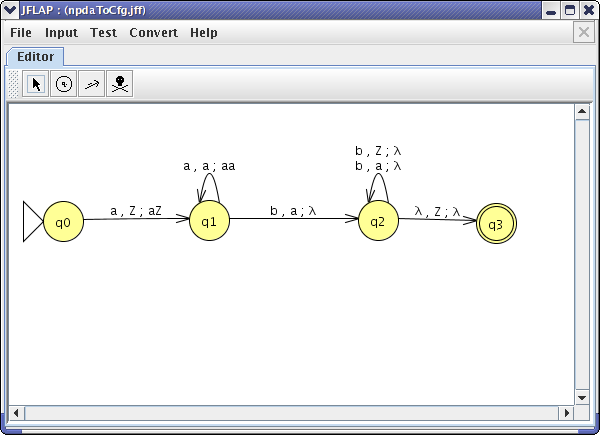
**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Pumping lemma button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

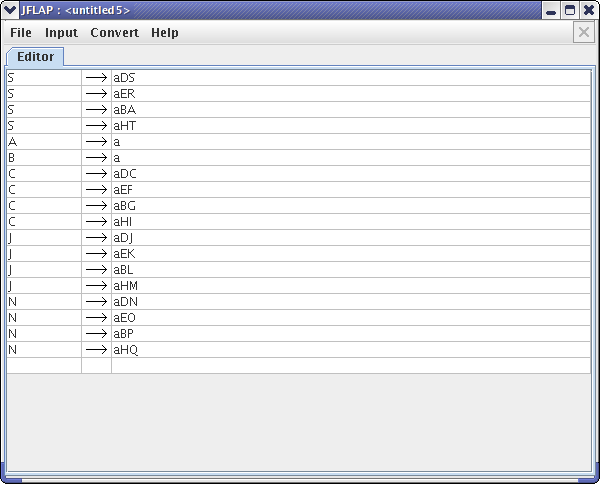
Screenshots:

1.

Input:



Output:



**RESULT:**

Thus, the given CFG from PDA is constructed successfully.

**EX.NO:9**

**CHOMSKY NORMAL FORM**

**AIM :** To construct Chomsky Normal Form for the following

S->ABC/BB

A->aA/Bac/aaa

B->bBb/a/D

C->CA/AC

**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Turing machine button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**SOLUTION:**

Step 1 : Eliminate € Production

s->ABC/BAB/AC/Ba/aB

A->aA/Bac/aaa/ac

B->bBb/a/bb/D

C->CA/AC

Step 2 :Eliminate unit productions

s->ABC/BAB/AC/Ba/aB

A->aA/Bac/aaa/ac

B->bBb/a/bb/D

C->CA/AC

Here D is unit production but is does not phase any further generation

Step 3: Eliminating useless symbols

Generating: D alone is not generating

Reaching: Here c is not reaching

The production is as follows

s->BaB/Ba/aB/a

A->aA/aaa

B->bBb/a/bb

Step 4 **: Chomsky Normal Form**

S ->BaB S -> Ba S ->aB B1 ->BBb

S ->BBbB S ->BBb S ->BbB Bb -> a

S -> B1B Bb -> a Bb -> a

A -> Aa A ->aaa B ->bBb B -> bbBc->b

A ->BbA A->BaBbBb B ->BcBBc B -> B2BcB->BCBc

A -> B2Bb B -> B3B Bc ->b

Bb -> a

B2 ->BbBb

S -> B1B | BBb |BbB | a

Bb -> a

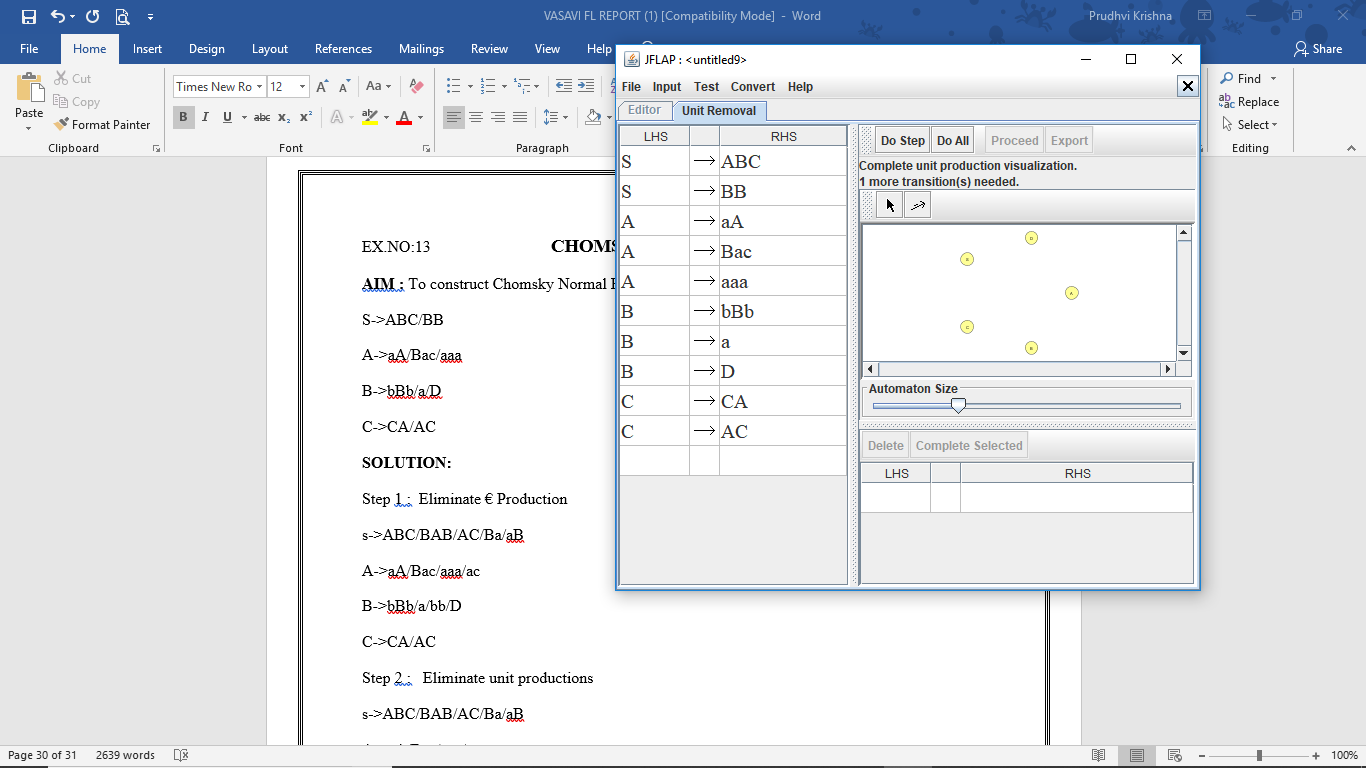
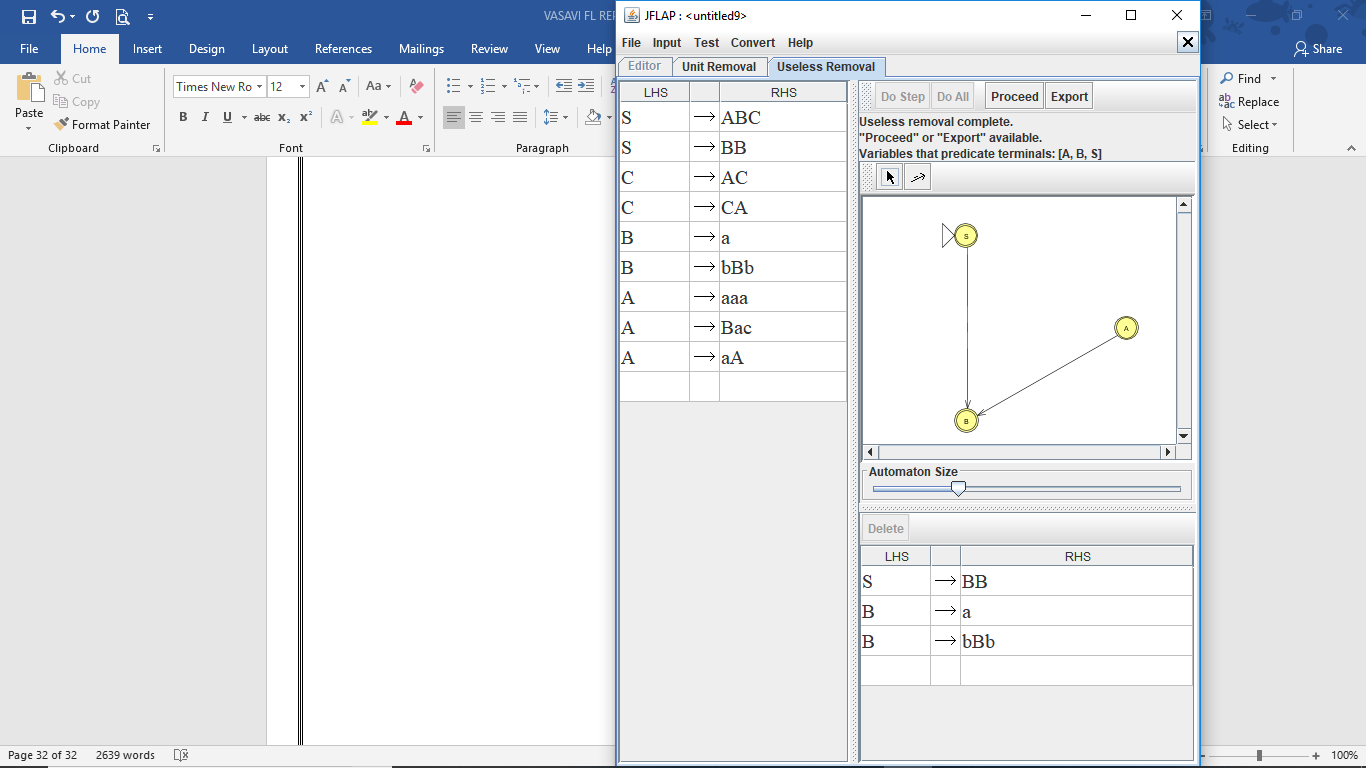
B1 -> Bb

A ->BbA | B2Bb

BbB0 -> B2

B -> B3B | BCBC | a

**SCREENSHOTS:**

**RESULT:**Thus,Chomsky Normal Form successfully verified.

**EX NO: 10**

**TURING MACHINE**

**AIM:**

To construct a Turing Machine that accepts anbn | n > 1**.**

.

**ALGORITHM:**

* Start the JFLAP8\_beta.jar file.
* Add required states.
* Make required transitions.
* Denote starting and final state.
* Save the project.
* Stop the program.

**PROCEDURE:**

* Start the JFLAP8\_beta.jar file.
* Click Turing machine button.
* Add required states to the Automaton editor.
* Make required transitions as per the problem.
* Denote starting and final state.
* Save the project.
* Test the automaton for few input values.
* Stop the program.

**TURING MACHINE REPRESENTATION:**

TM M={ Q,∑, ˧, ᵟ ,q0,B,F}

BB q0aabbBB

ᵟ(q0**,**a) = (q1 , X ,R) BBXq1abbBB

ᵟ(q1 **,**a) = (q1 , a ,R) BBXaq1 bbBB

ᵟ(q1 **,**b) = (q2 , Y ,L) BBXaq2 YbBB

ᵟ(q2 **,**a) = (q2 , a ,L) BBXq2 aYbBB

ᵟ(q2 **,** X) = (q0 , X ,R) BBXq0 aYbBB

ᵟ(q0**,**a) = (q1 , X ,R) BBXXq1 YbBB

ᵟ(q1 **,**Y) = (q1 , Y ,R) BBXXYq1 YBB

ᵟ(q1 **,**b) = (q2 , Y ,L) BBXXYq2 YBB

ᵟ(q1 **,**Y) = (q2 , Y ,L) BBXXq2YYBB

ᵟ(q2 **,**X) = (q0 , X ,R) BBXXq0 YYBB

ᵟ(q0**,**Y) = (q3 , Y ,R) BBXXYq3 YBB

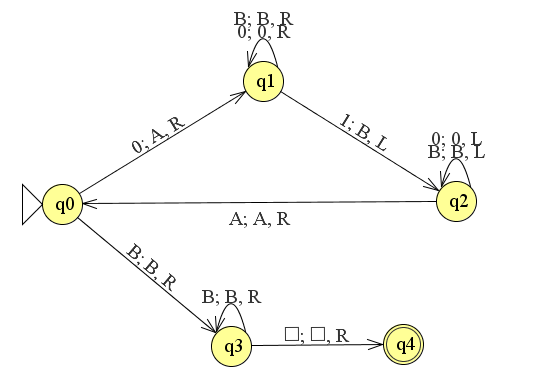
ᵟ(q3 **,**Y) = (q3 , Y ,R) BBXXYYq3BB

ᵟ(q3 **,**B) = (q4 , B ,R) BBXXYYBq4B

**SOLUTION:**

**TURING MACHINE TRANSITION:**

**SCREENSHOTS:**

****

**RESULT:** Thus, the given Turing Machine is designed successfully.