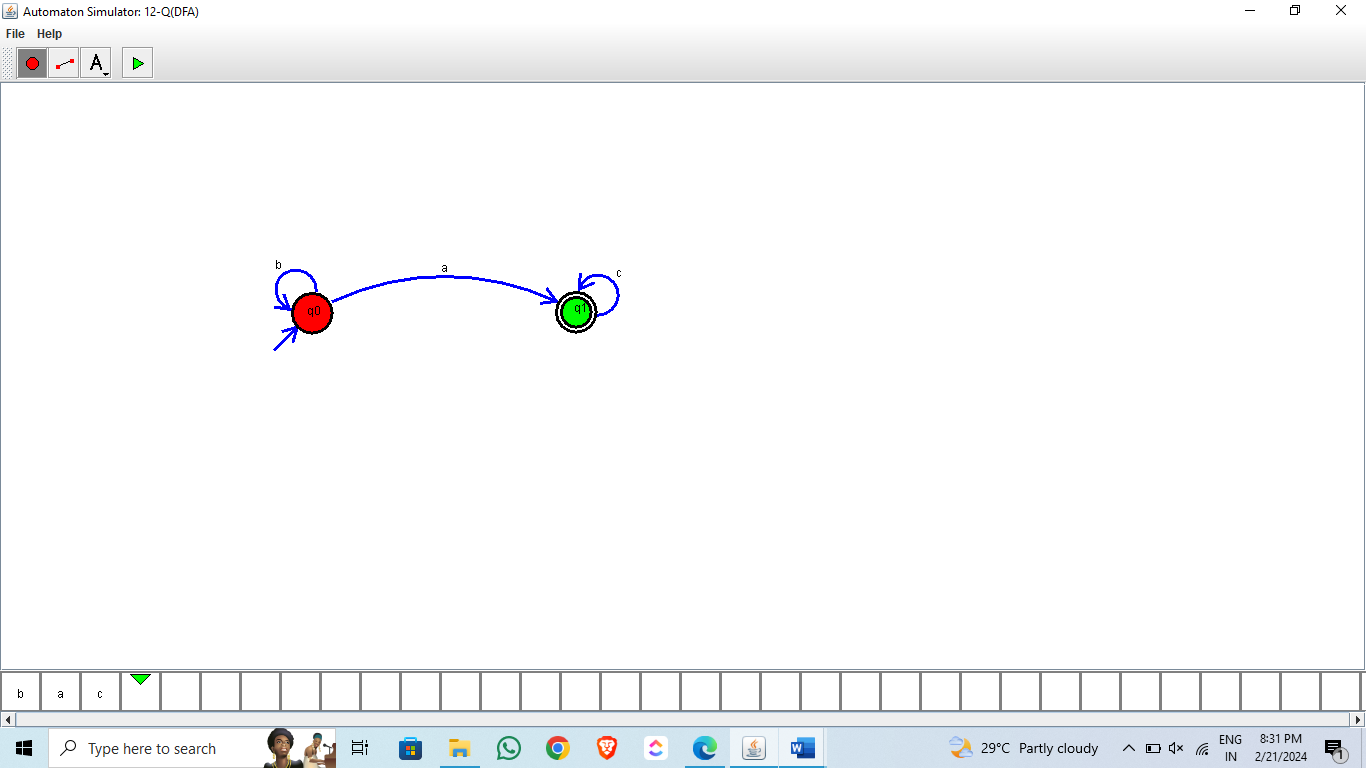
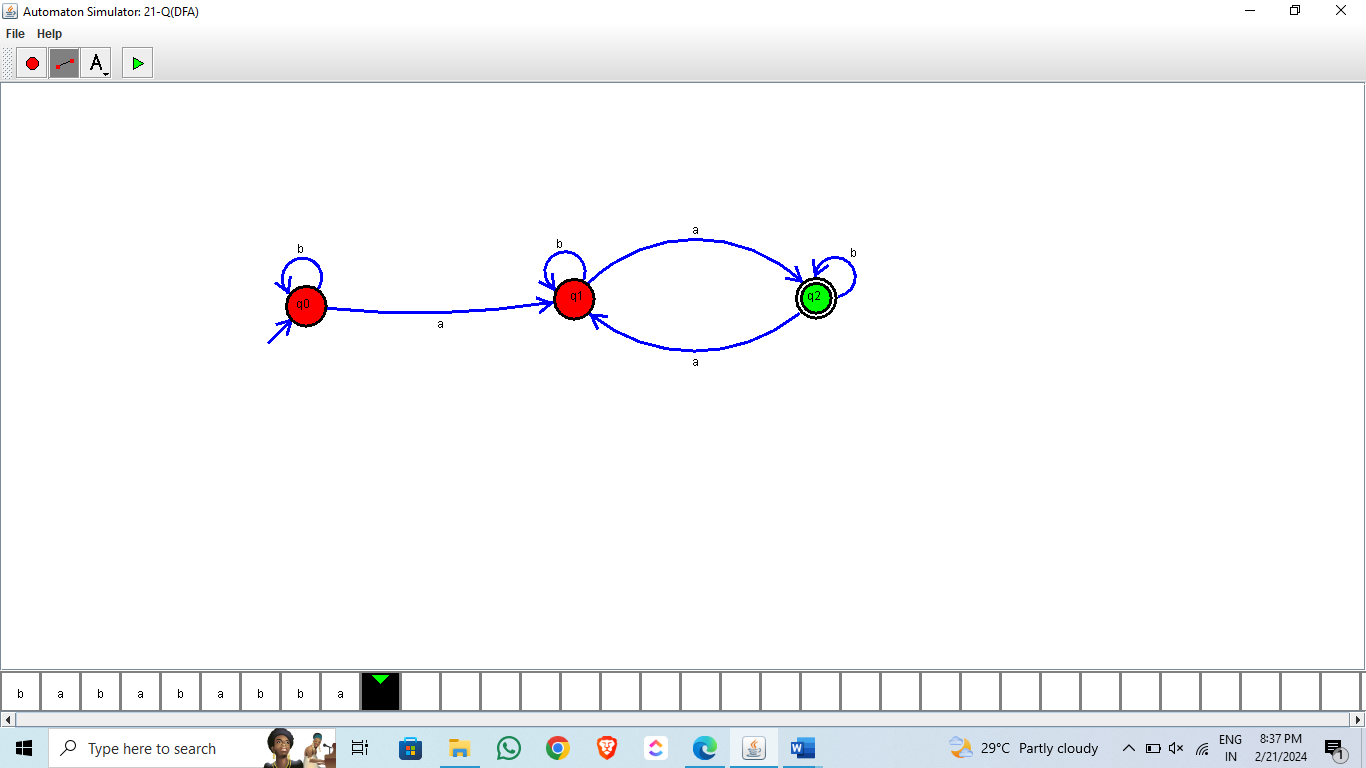
TOC-EXPERIMENTS

DFA:

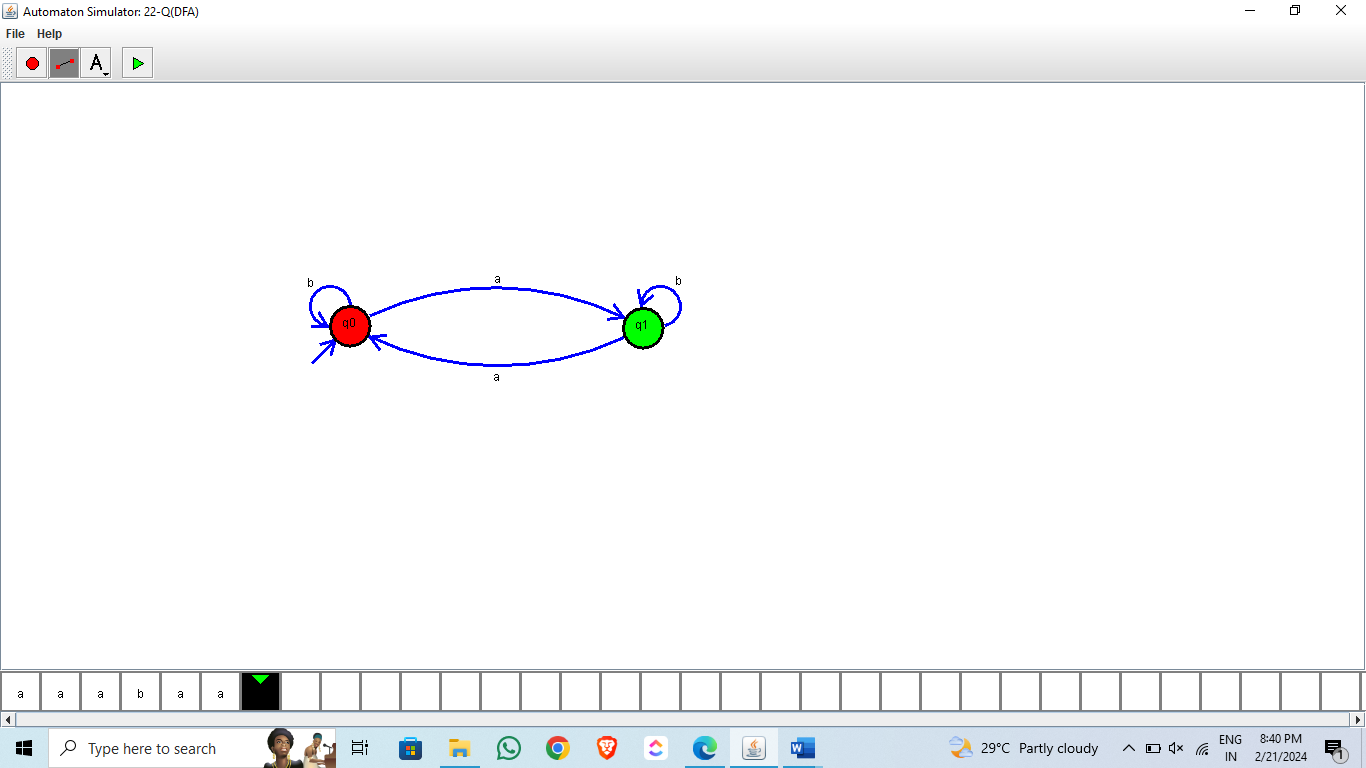
1. Design DFA using simulator to accept the input string “a” ,”ac”,and ”bac”.



1. Design DFA using simulator to accept even number of a’s.

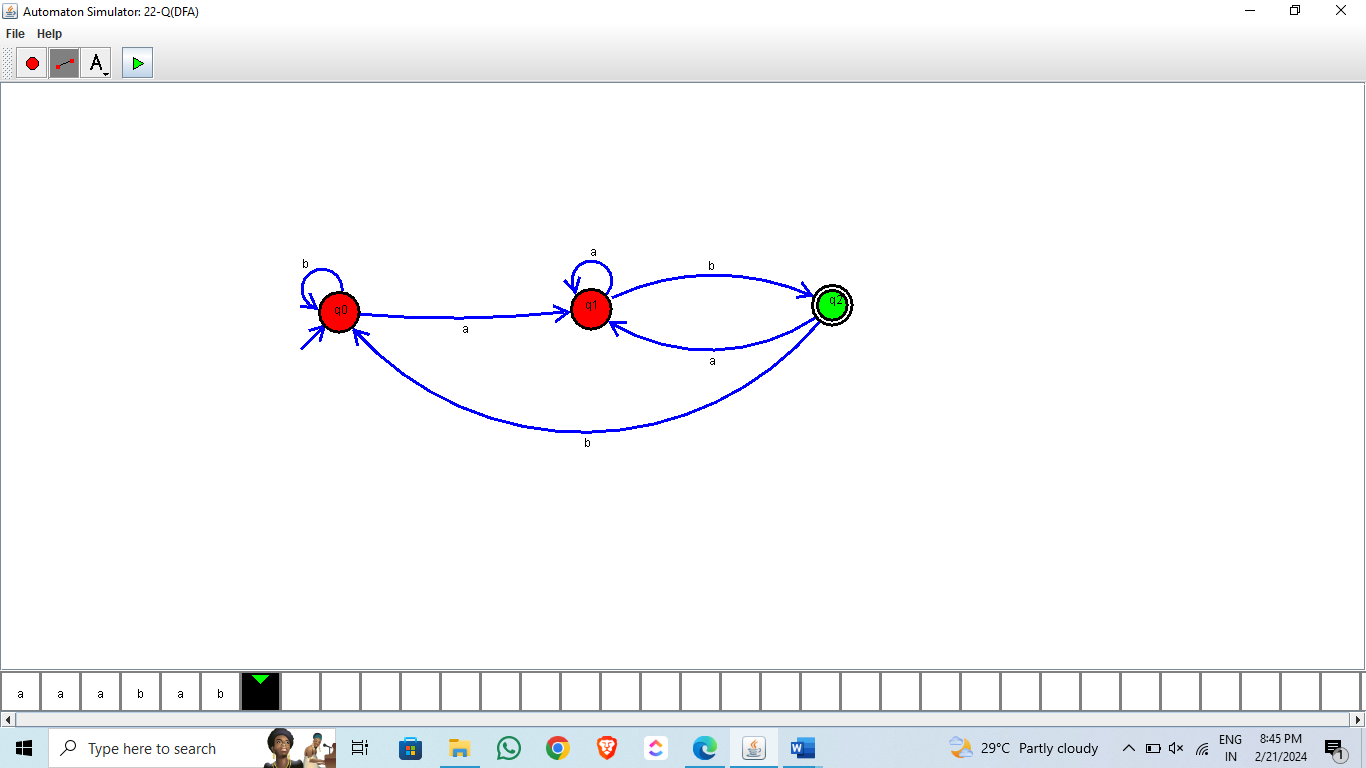


1. Design DFA using simulator to accept odd number of a’s

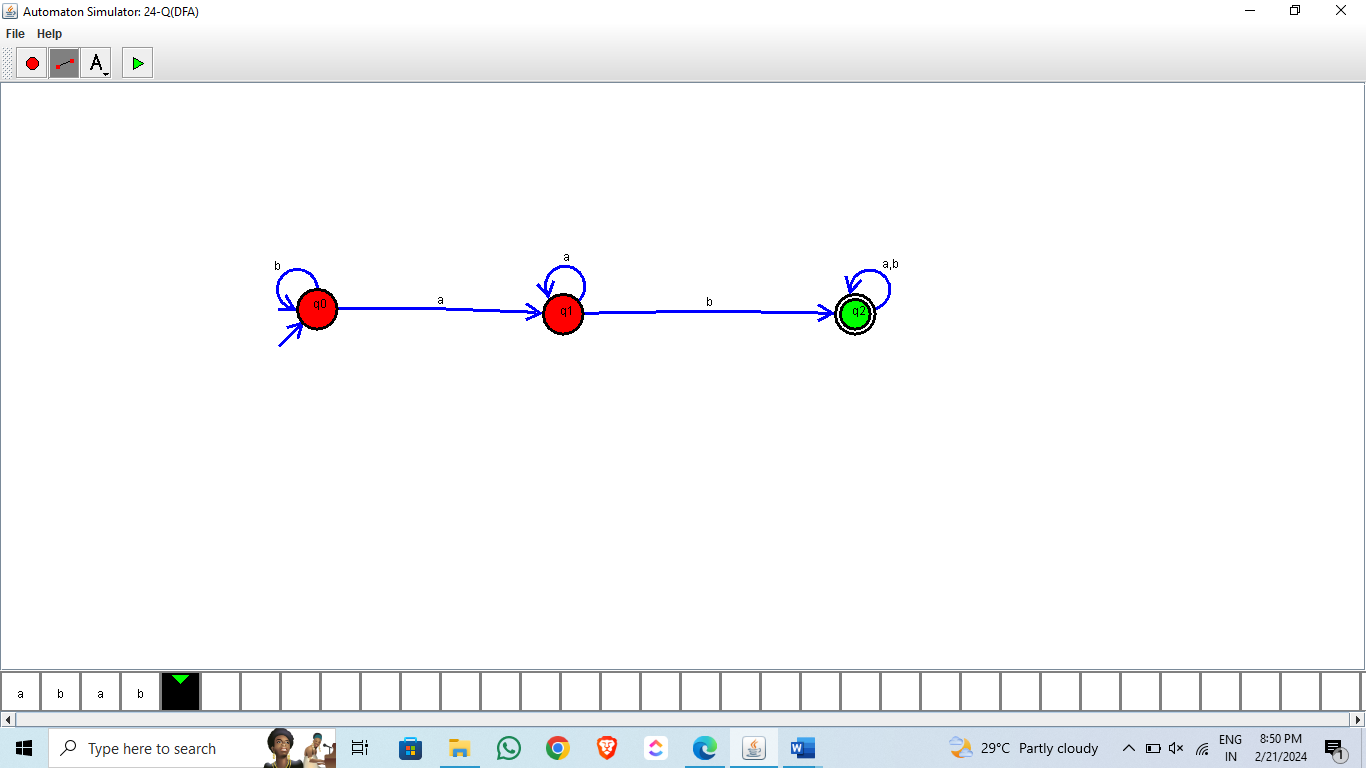


1. Design DFA using simulator to accept the string the end with ab over set {a,b)

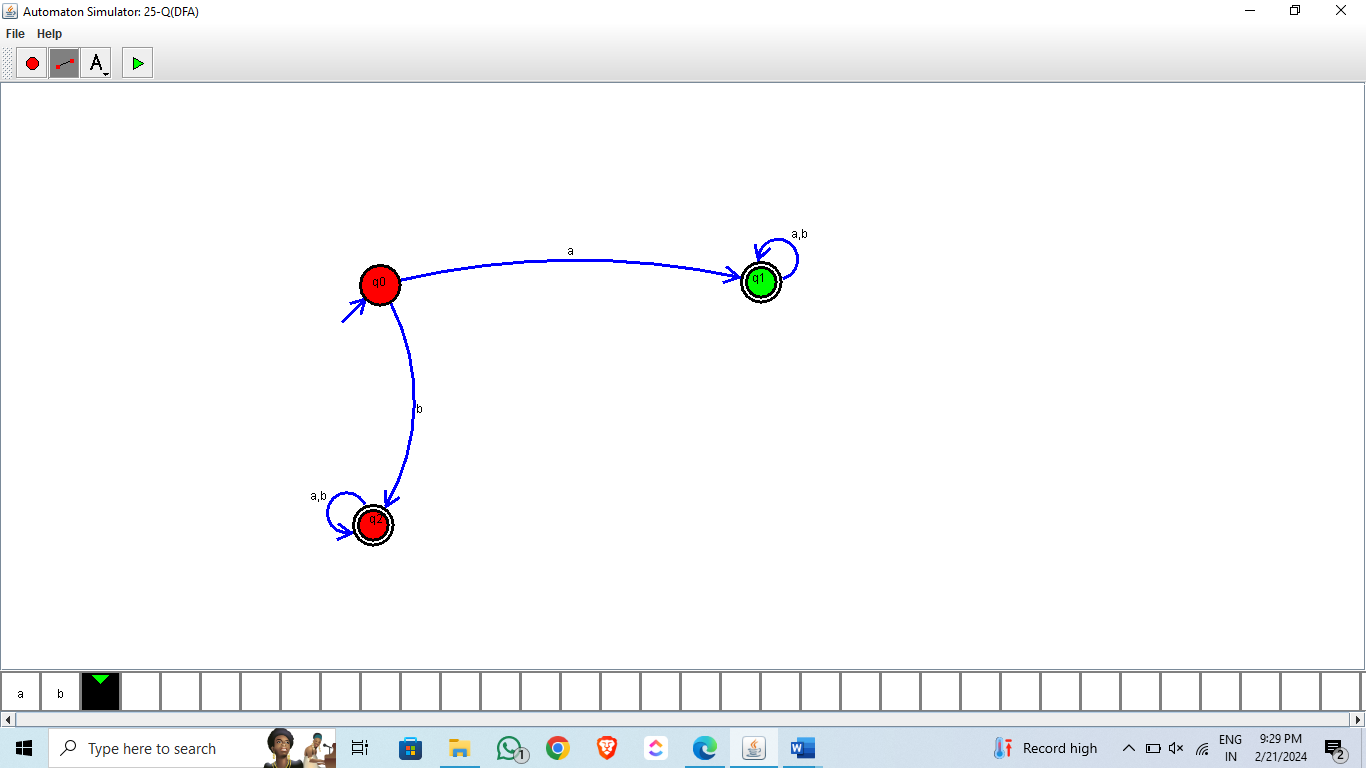
W= aaabab



1. Design DFA using simulator to accept the string having ‘ab’ as substring over the set {a,b}

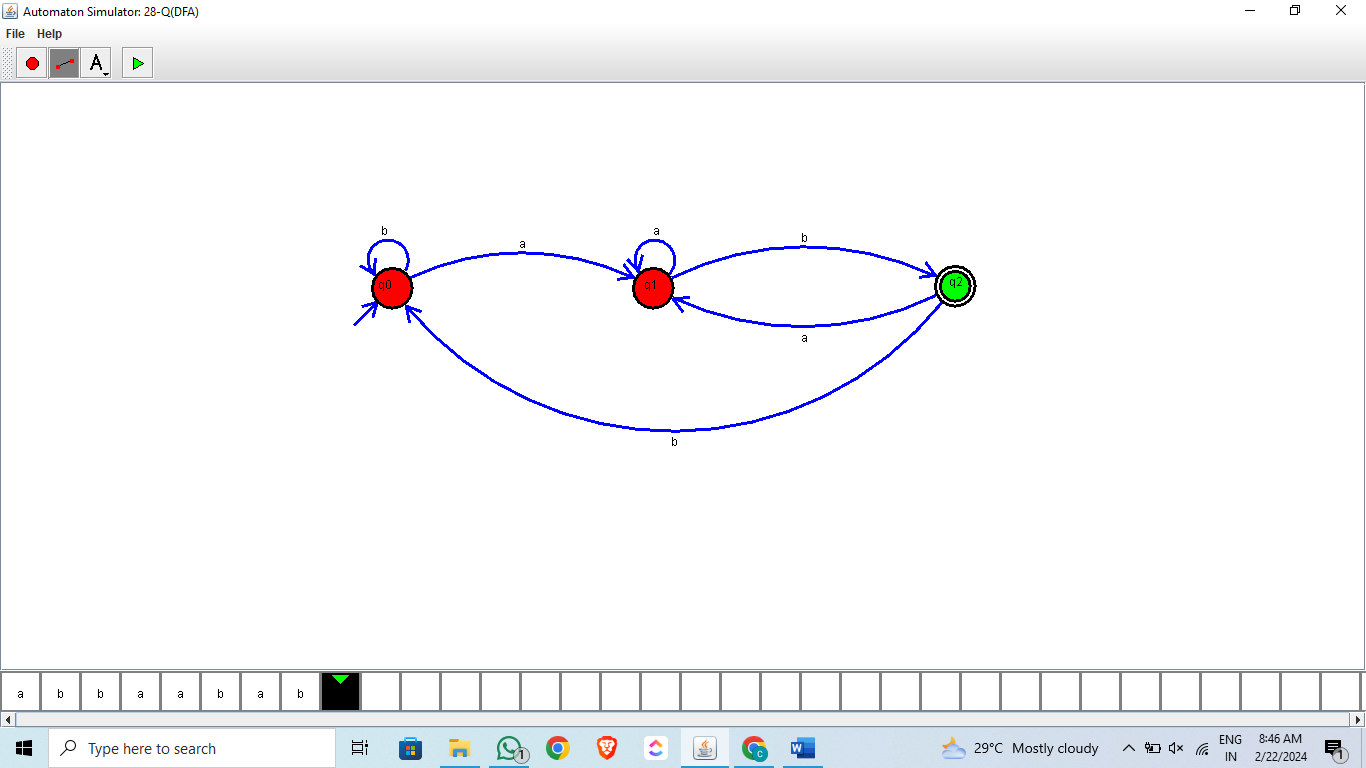


1. Design DFA using simulator to accept the string start with a or b over the set {a,b}

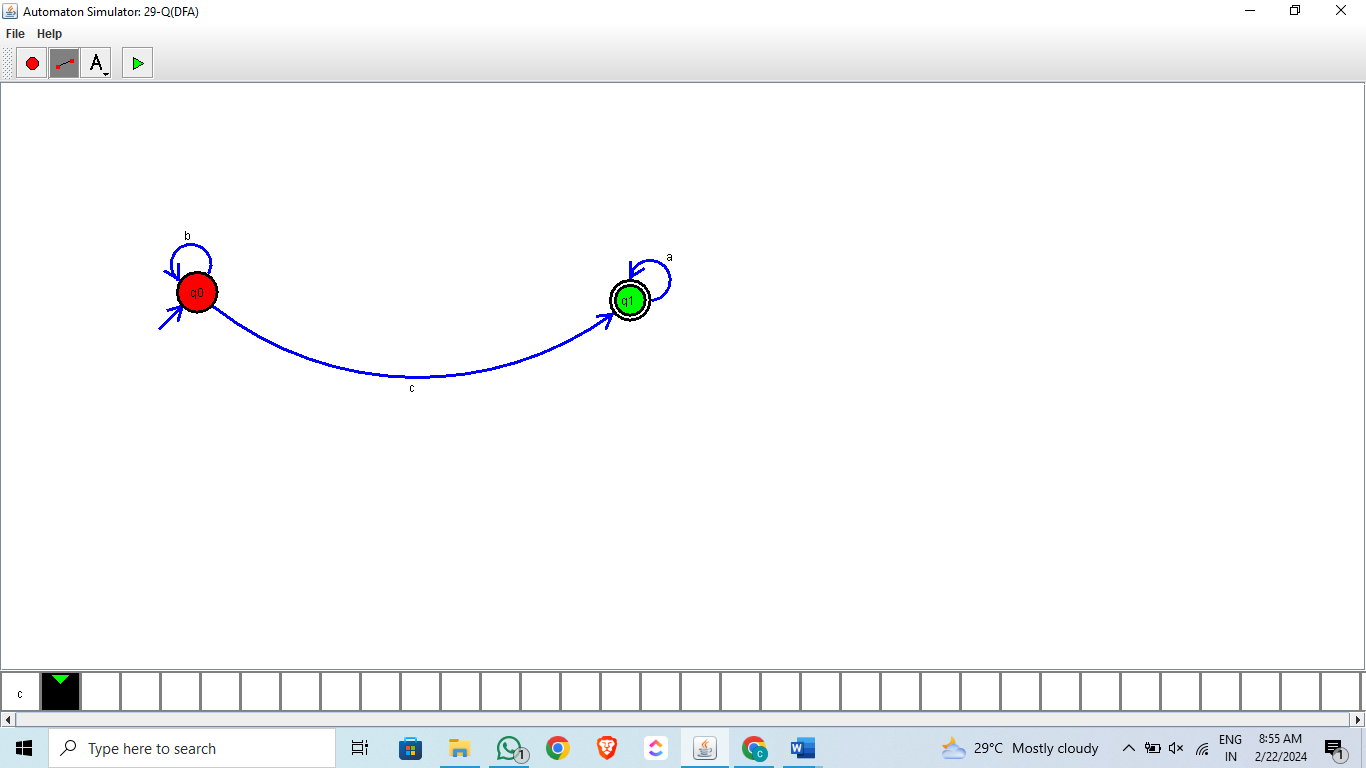


1. Design DFA using simulator to accept the string the end with ab over set {a,b)

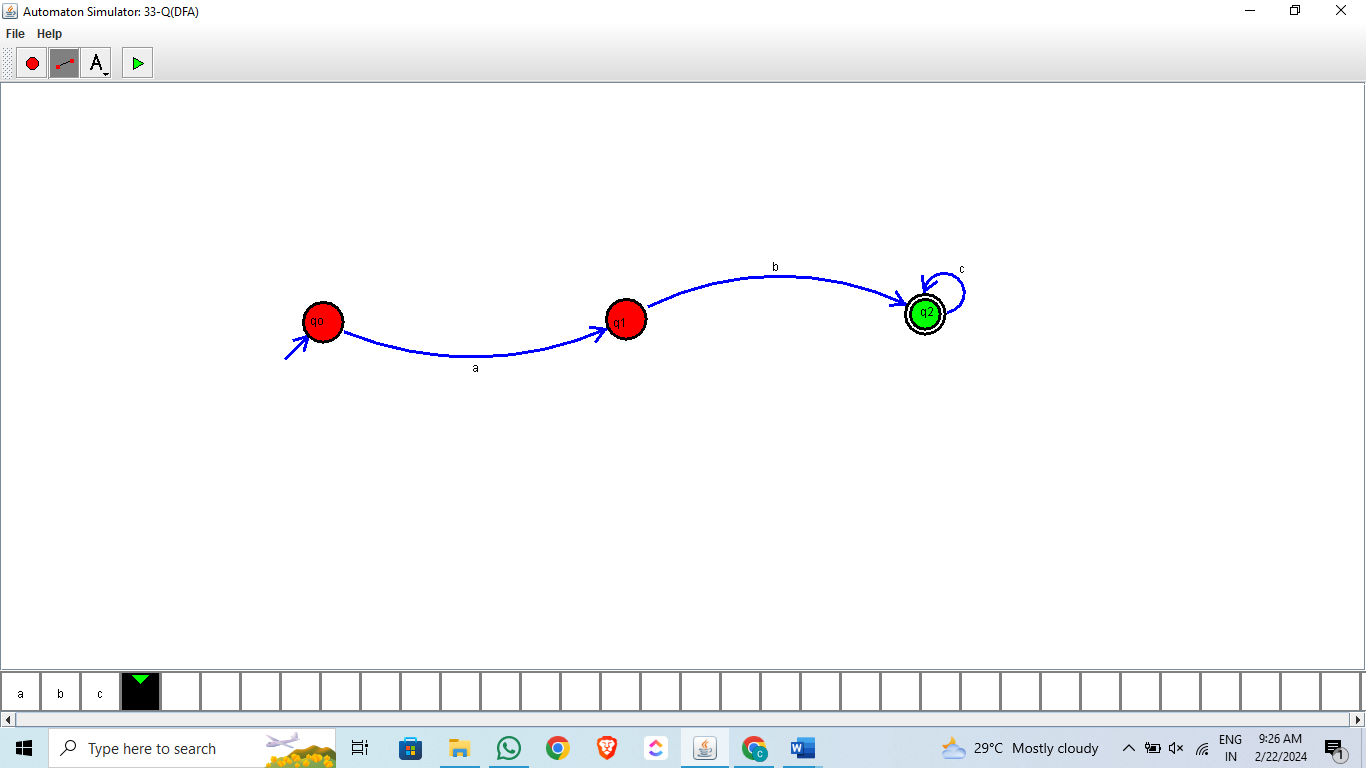
W= abbaabab



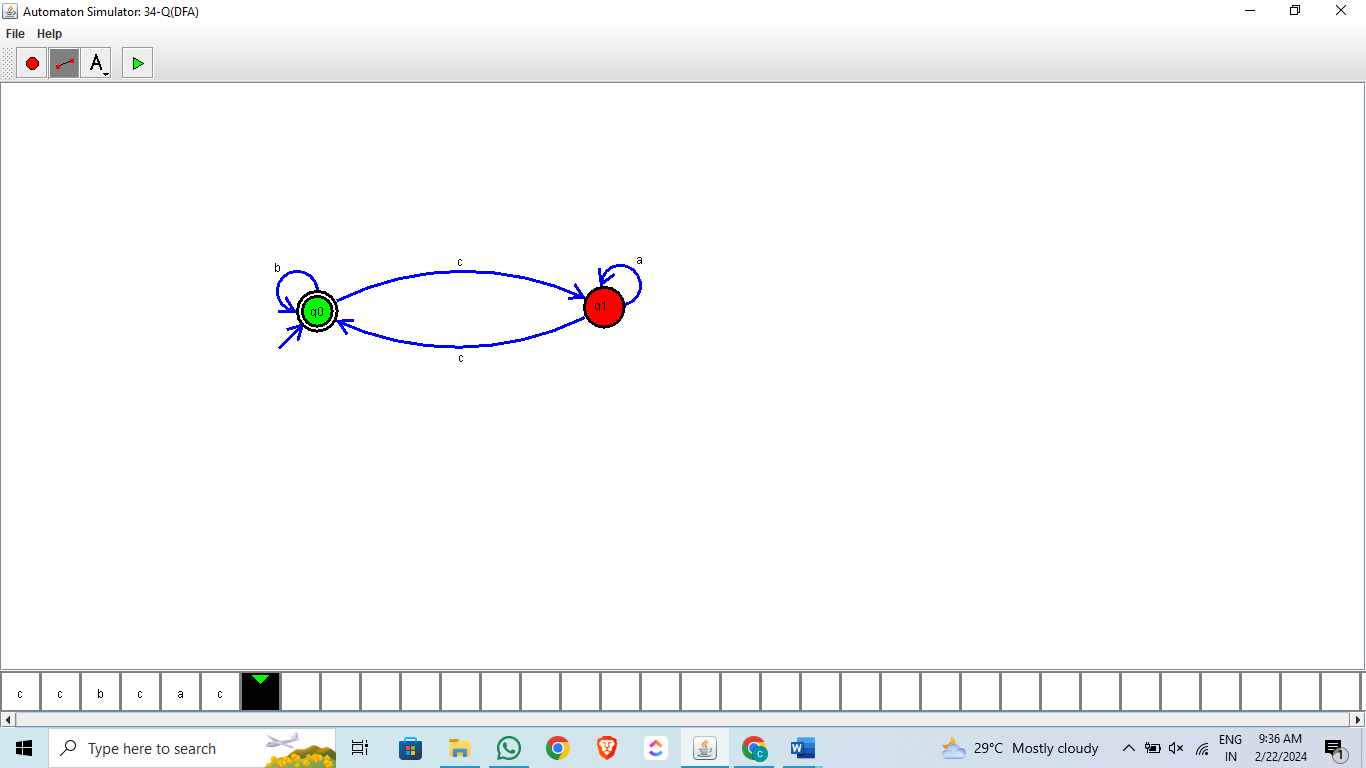
1. Design DFA using simulator to accept the input string “bc” ,”c”,and ”bcaaa”.



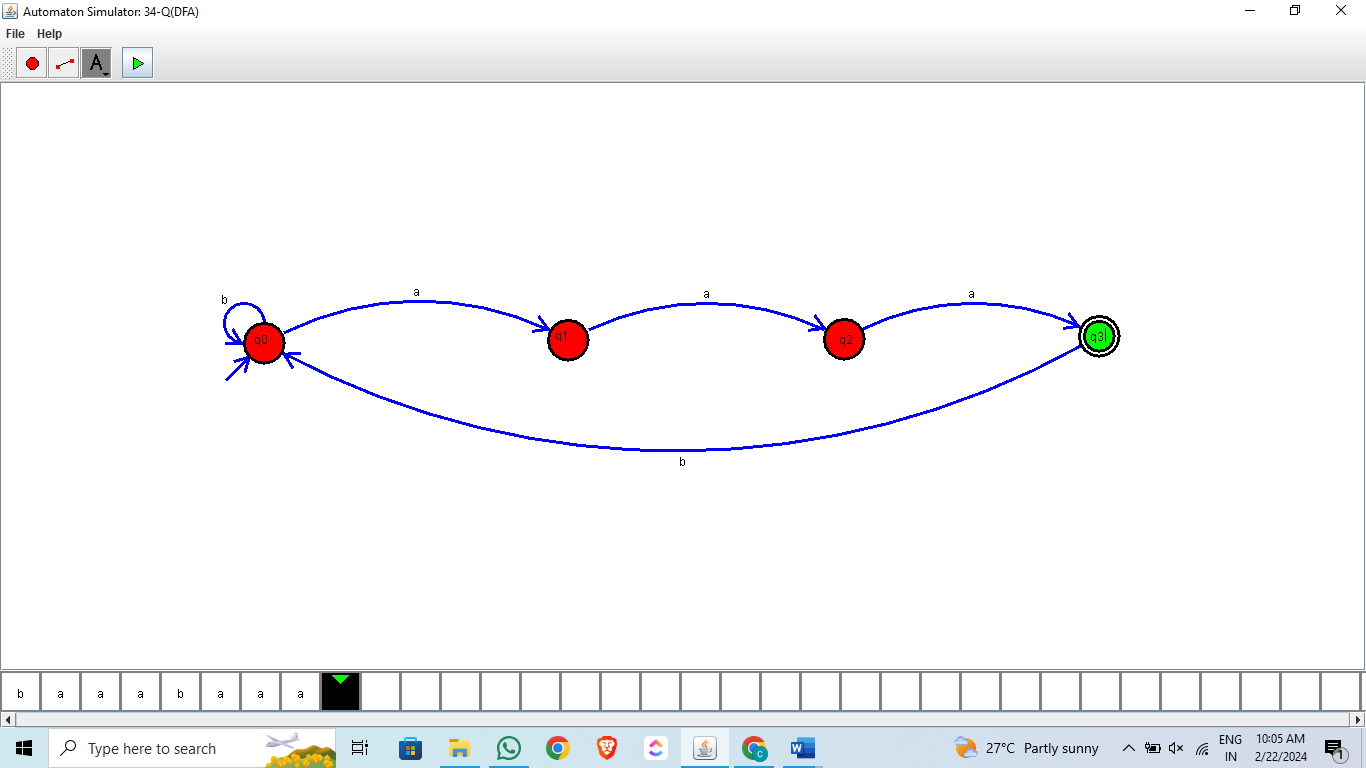
1. Design DFA using simulator to accept the string having ‘abc’ as substring over the set {a,b,c}



1. Design DFA using simulator to accept even number of c’s over the set {a,b,c}

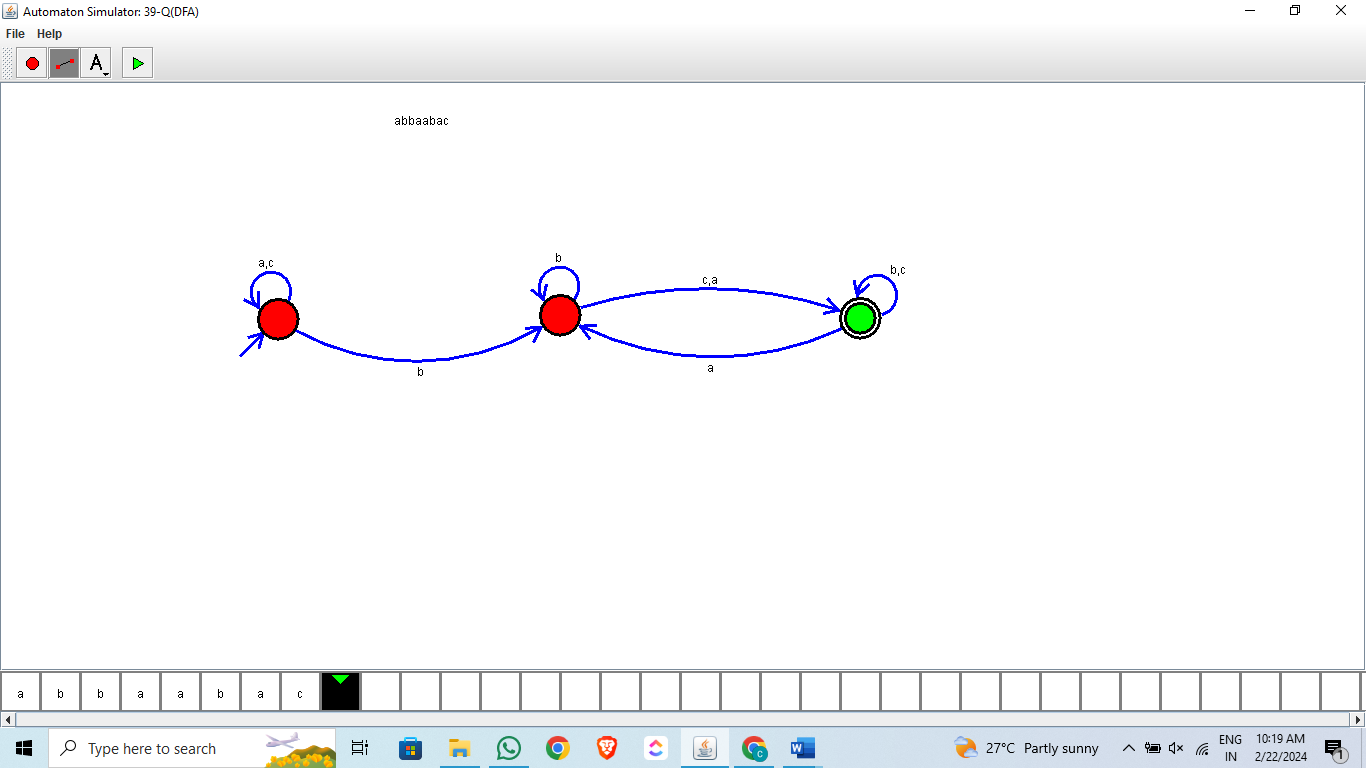


1. Design DFA using simulator to accept strings in which a’s always appear tripled over input {a,b}



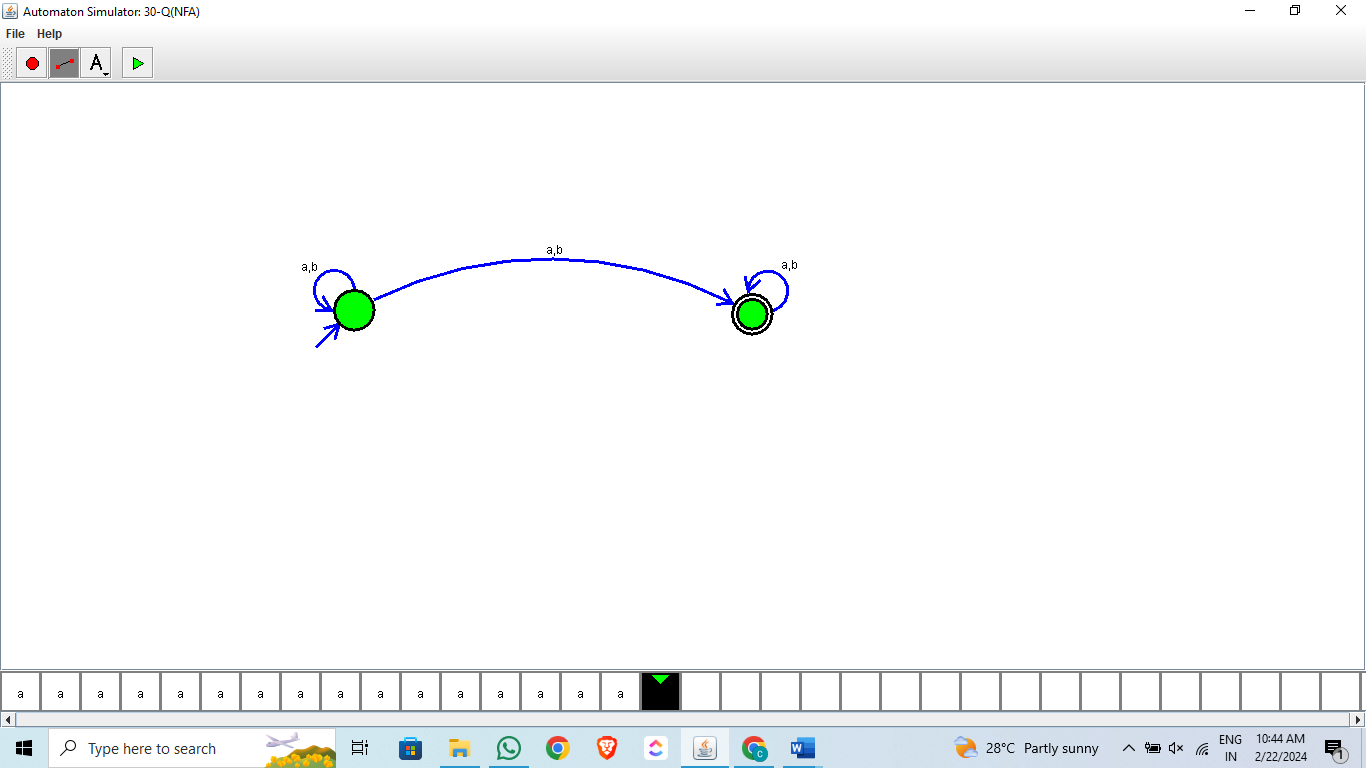
1. Design DFA using simulator to accept the string the end with abc over set {a,b,c)

W= abbaababc.

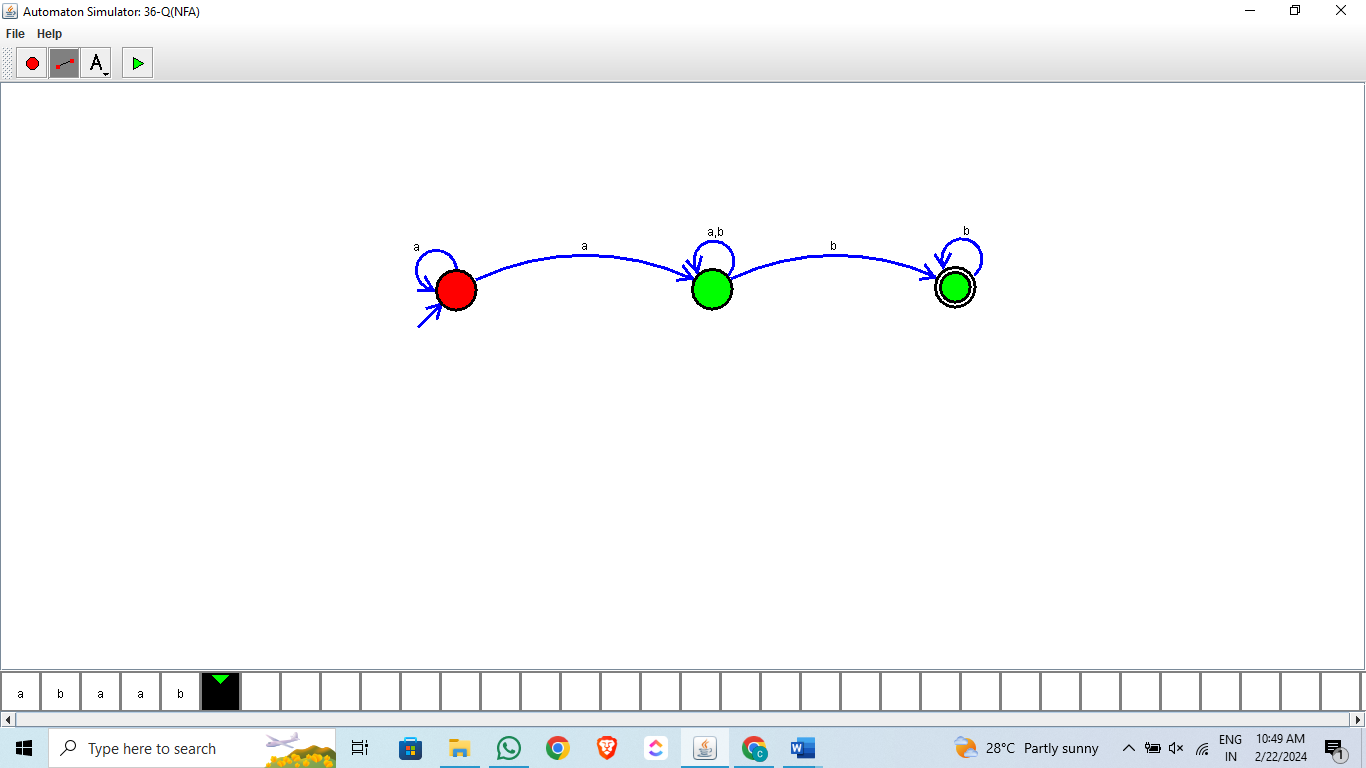


NFA:

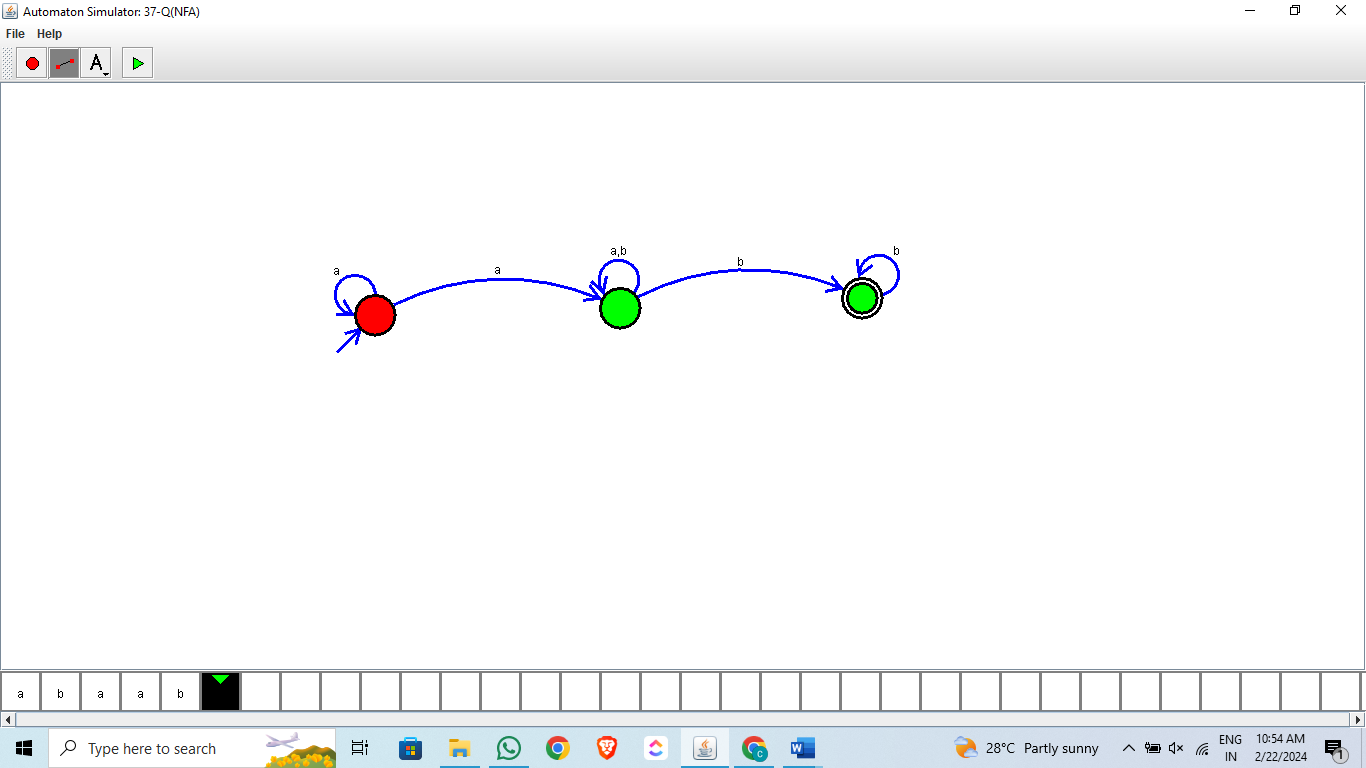
1. Design NFA to accept any number of a’s where input={a,b}.



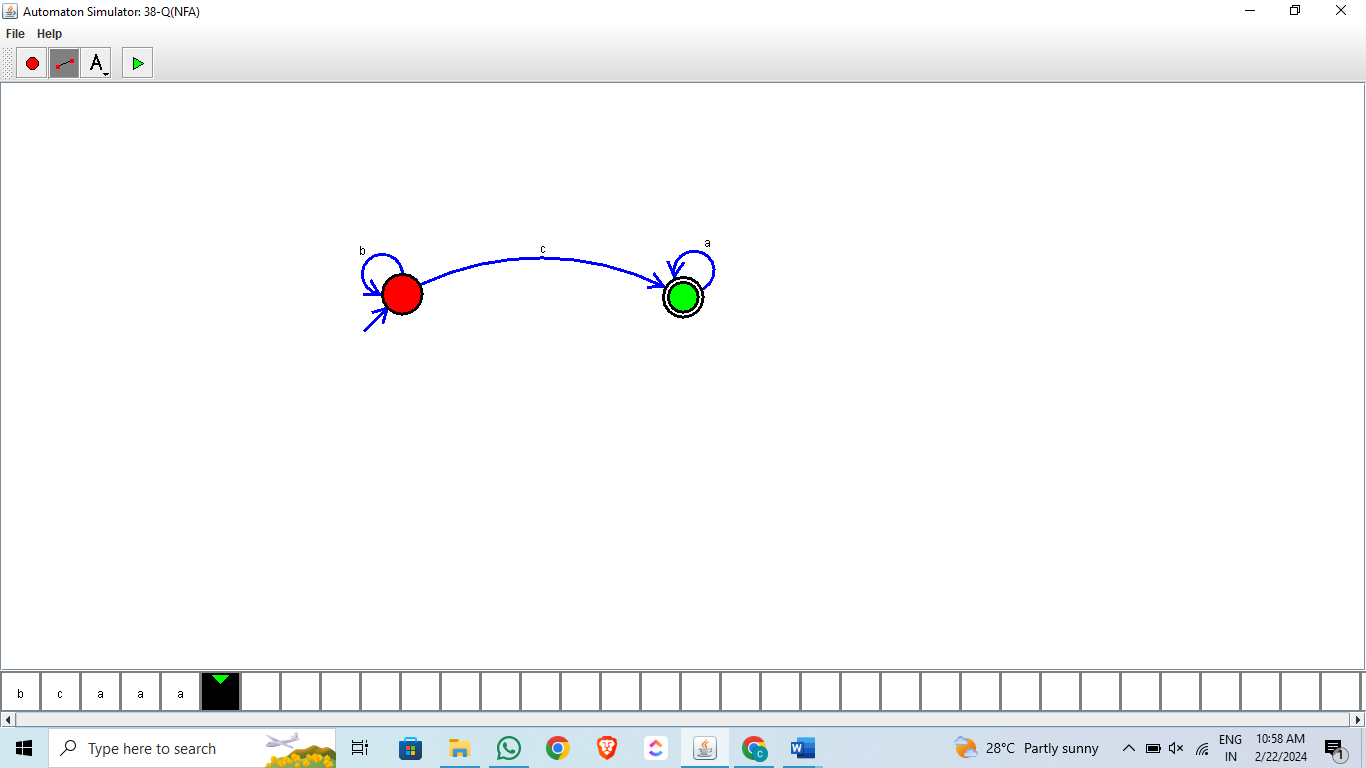
1. Design NFA using simulator to accept the string the start with a and end with b over set {a,b} and check W= abaab is accepted or not.



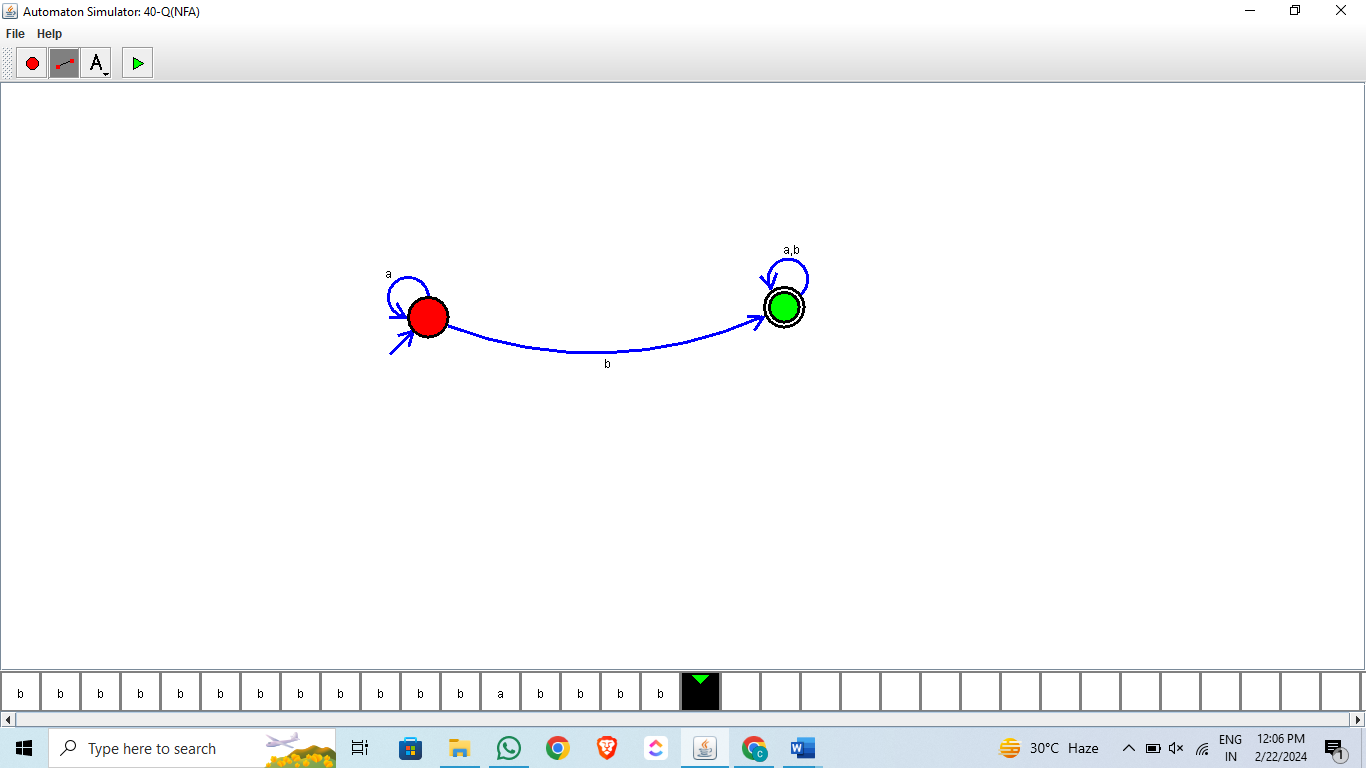
1. Design NFA using simulator to accept the string that start and end with different symbols over the input {a,b}.



1. Design NFA using simulator to accept the input string “bbc” ,”c”,and ”bcaaa”.

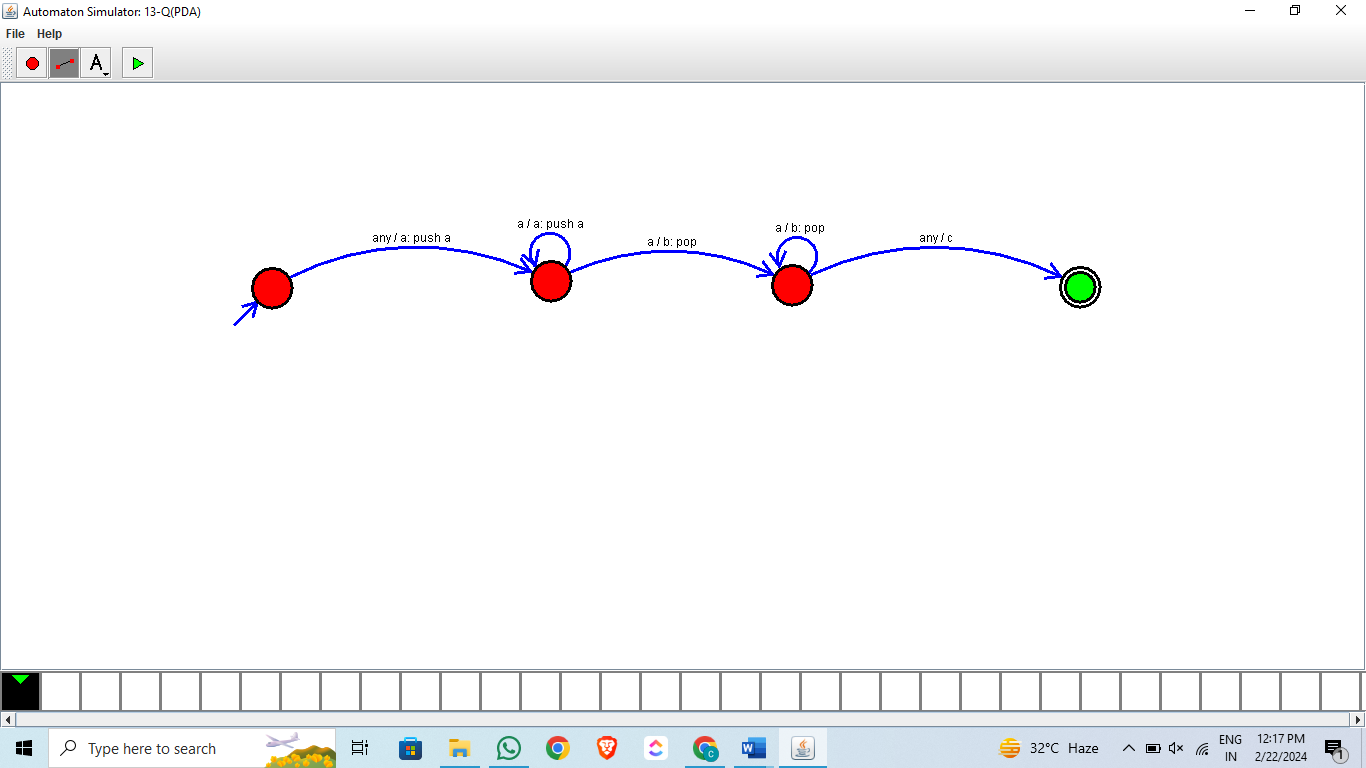


1. Design NFA to accept any number of b’s where input={a,b}.

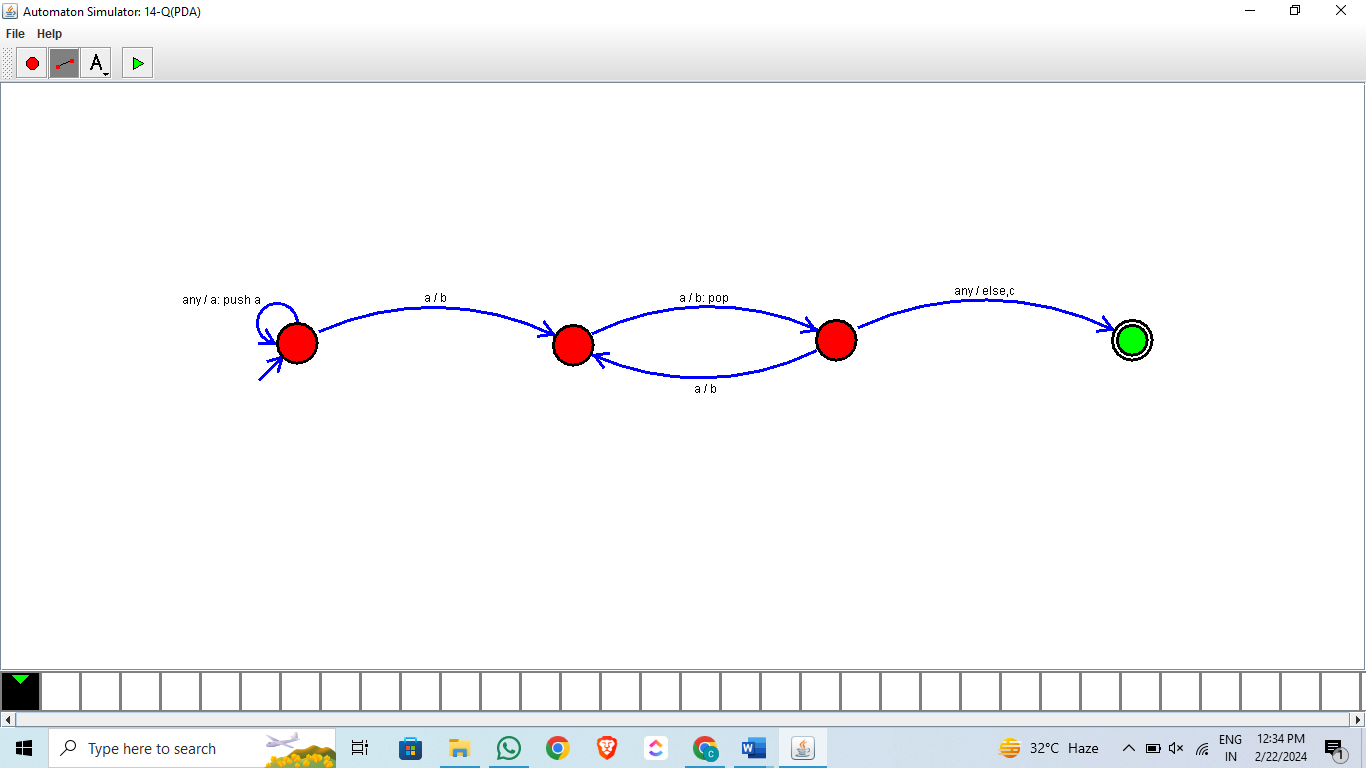


PDA:

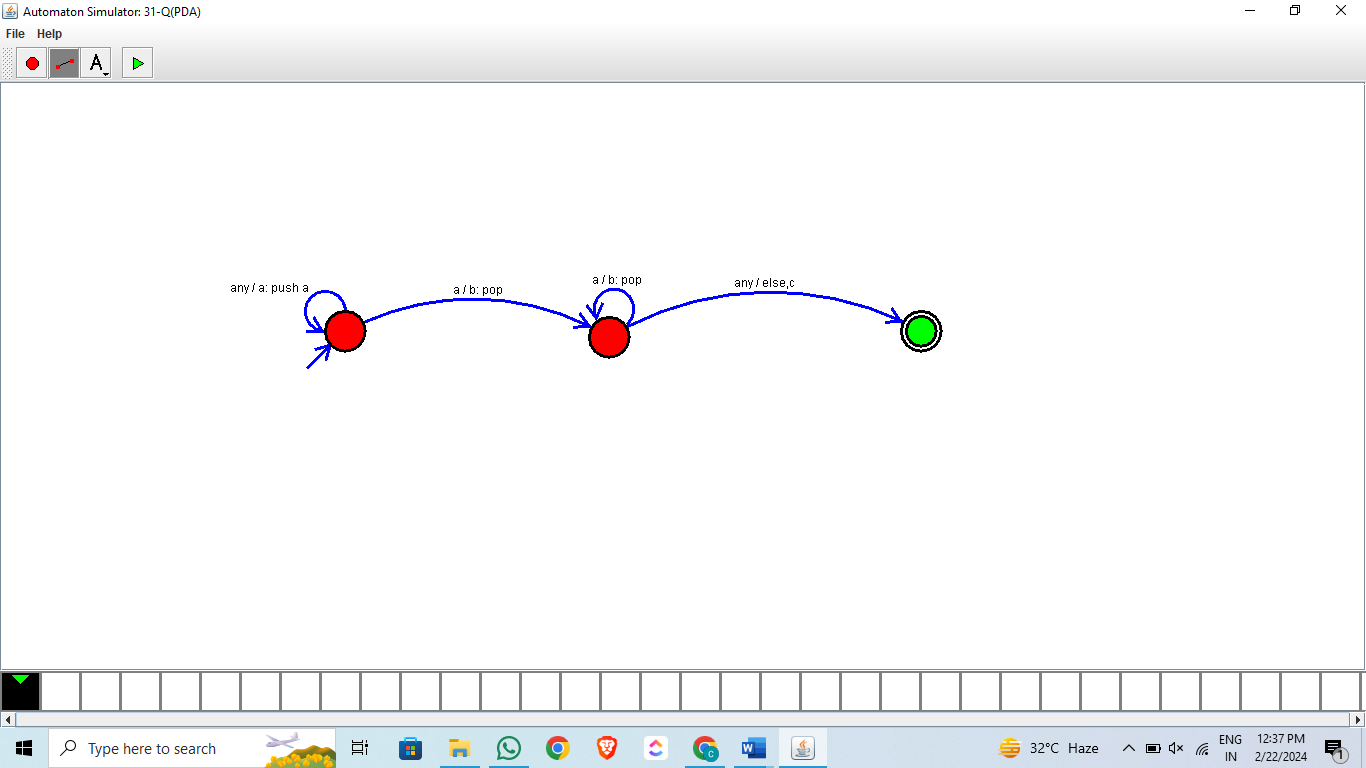
1. Design PDA using simulator to accept the input string aabb



1. Design PDA using simulator to accept the input string anb2n

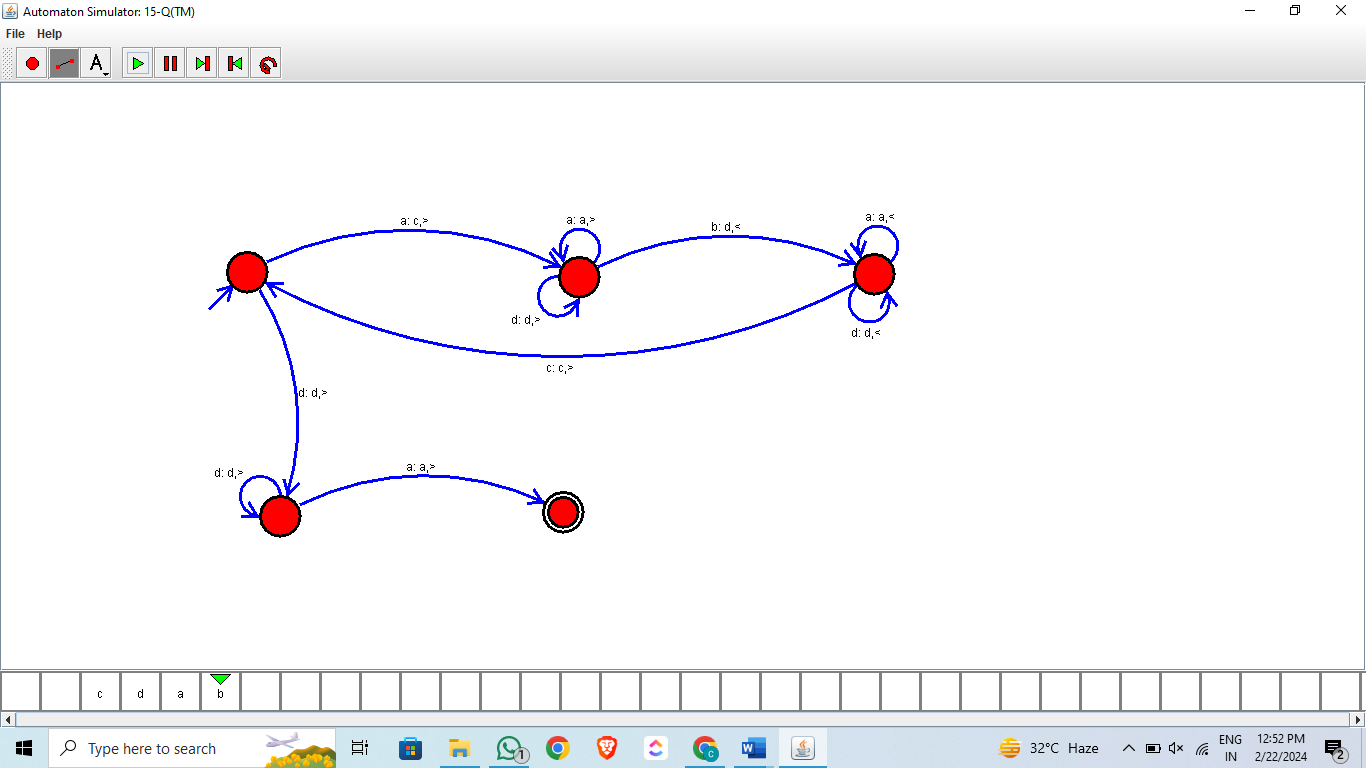


1. Design PDA using simulator to accept the input string anbn

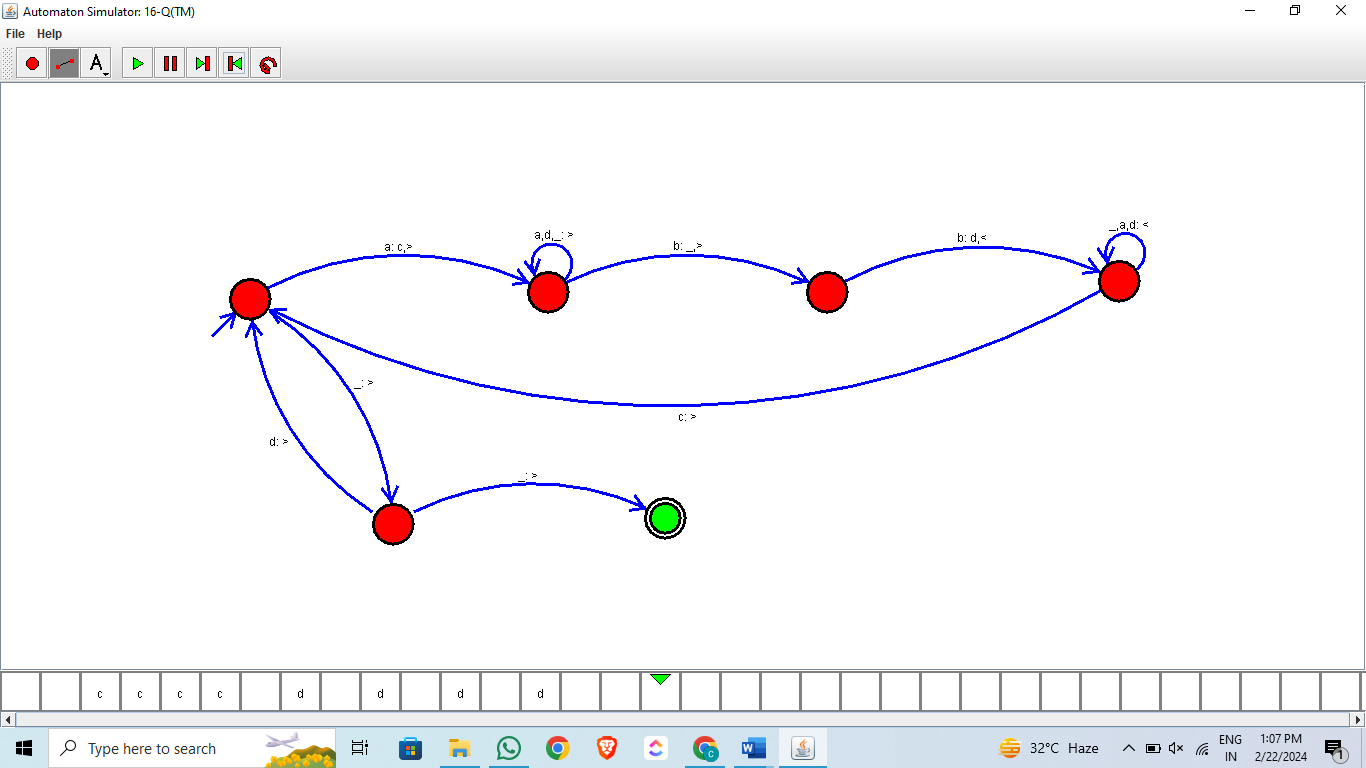


TM:

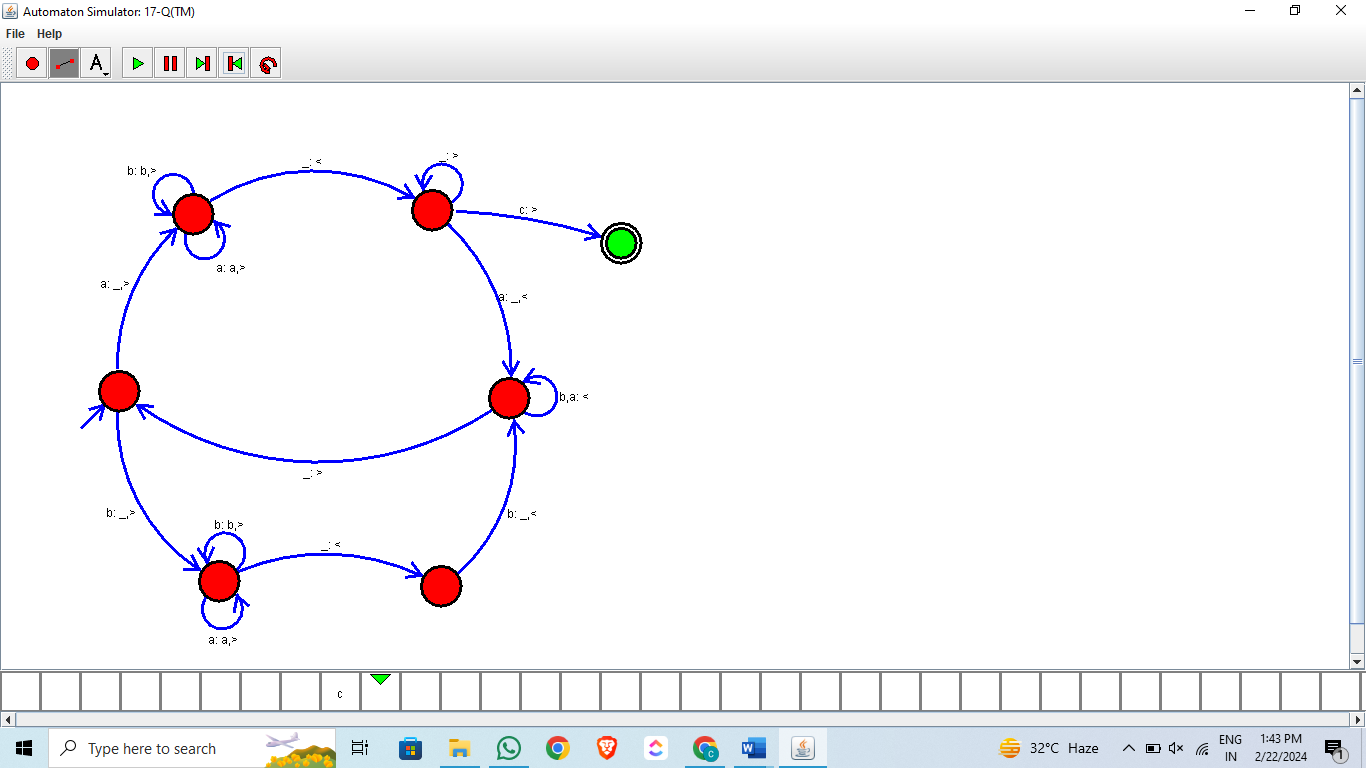
1. Design TM using simulator to accept the input string anbn



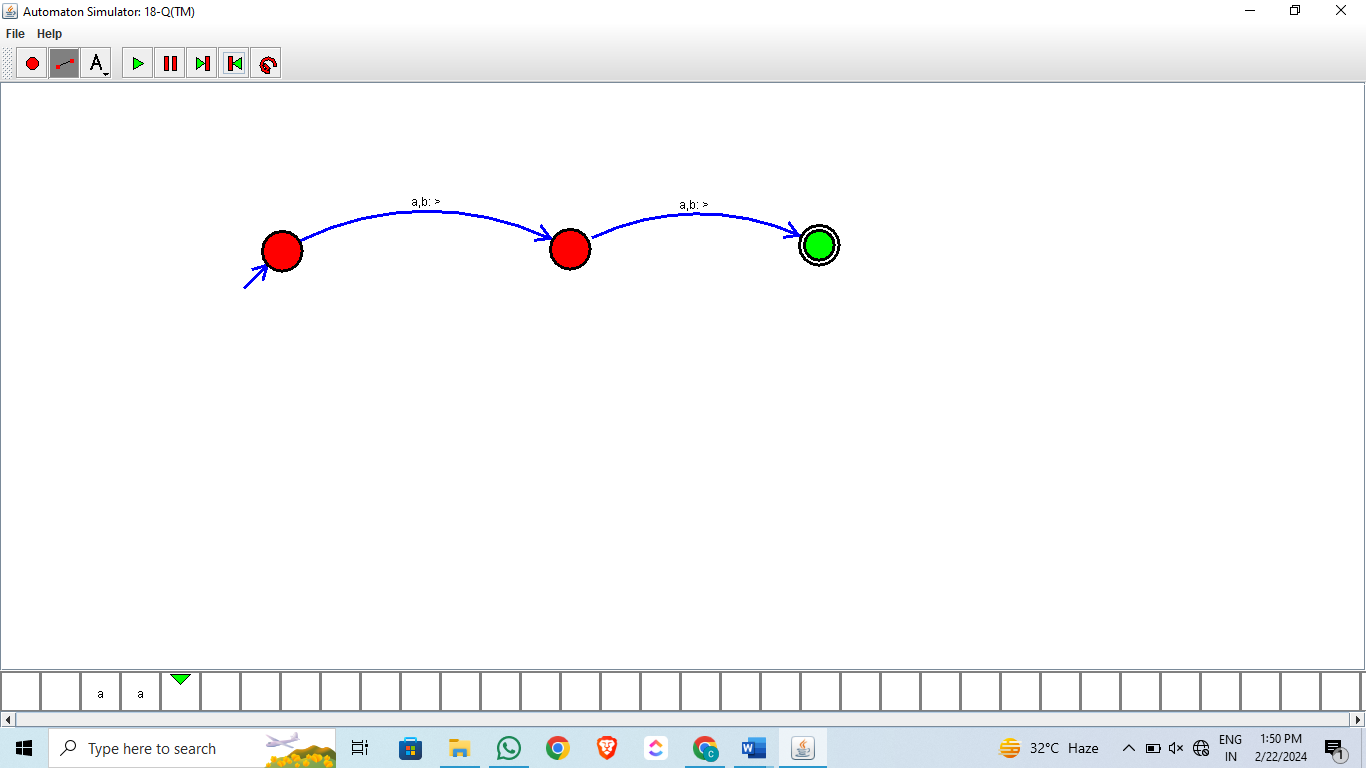
1. Design TM using simulator to accept the input string anb2n



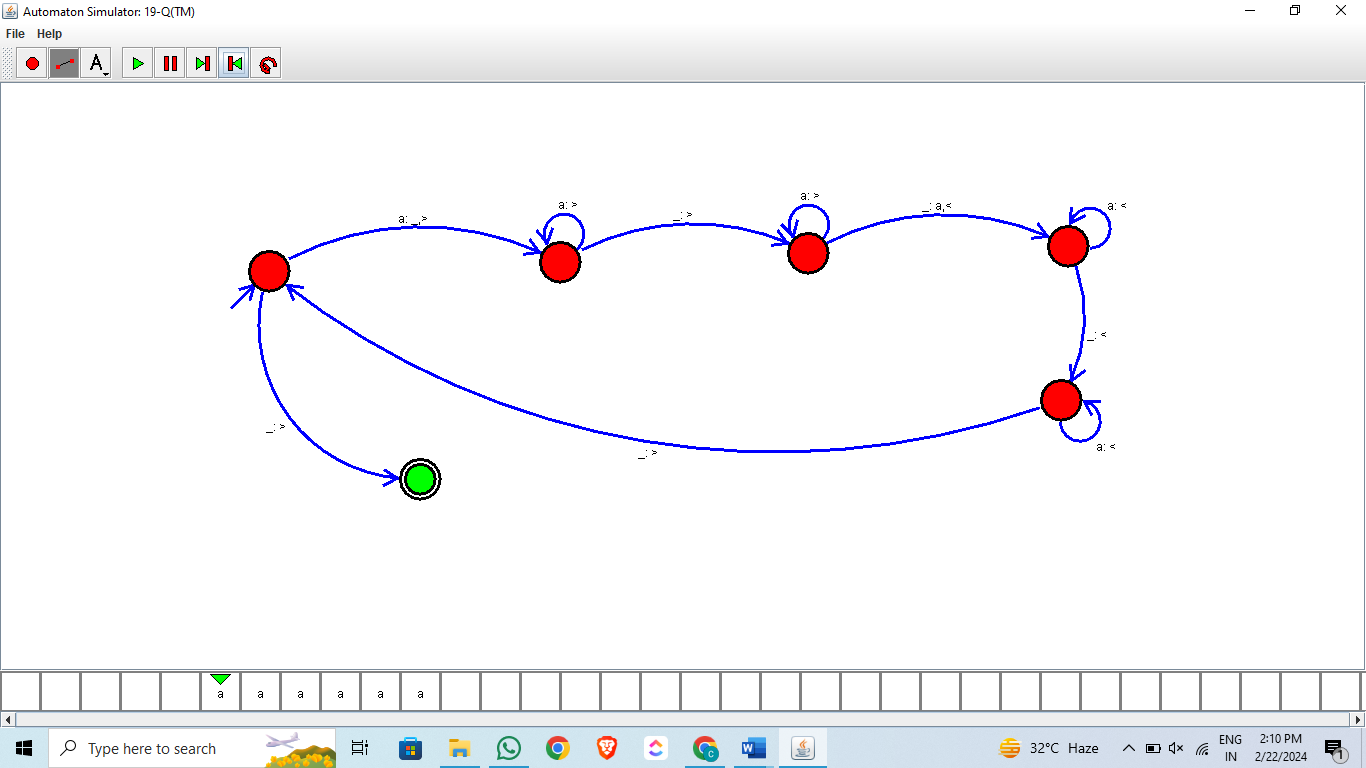
1. Design TM using simulator to accept the input string Palindrome ababa



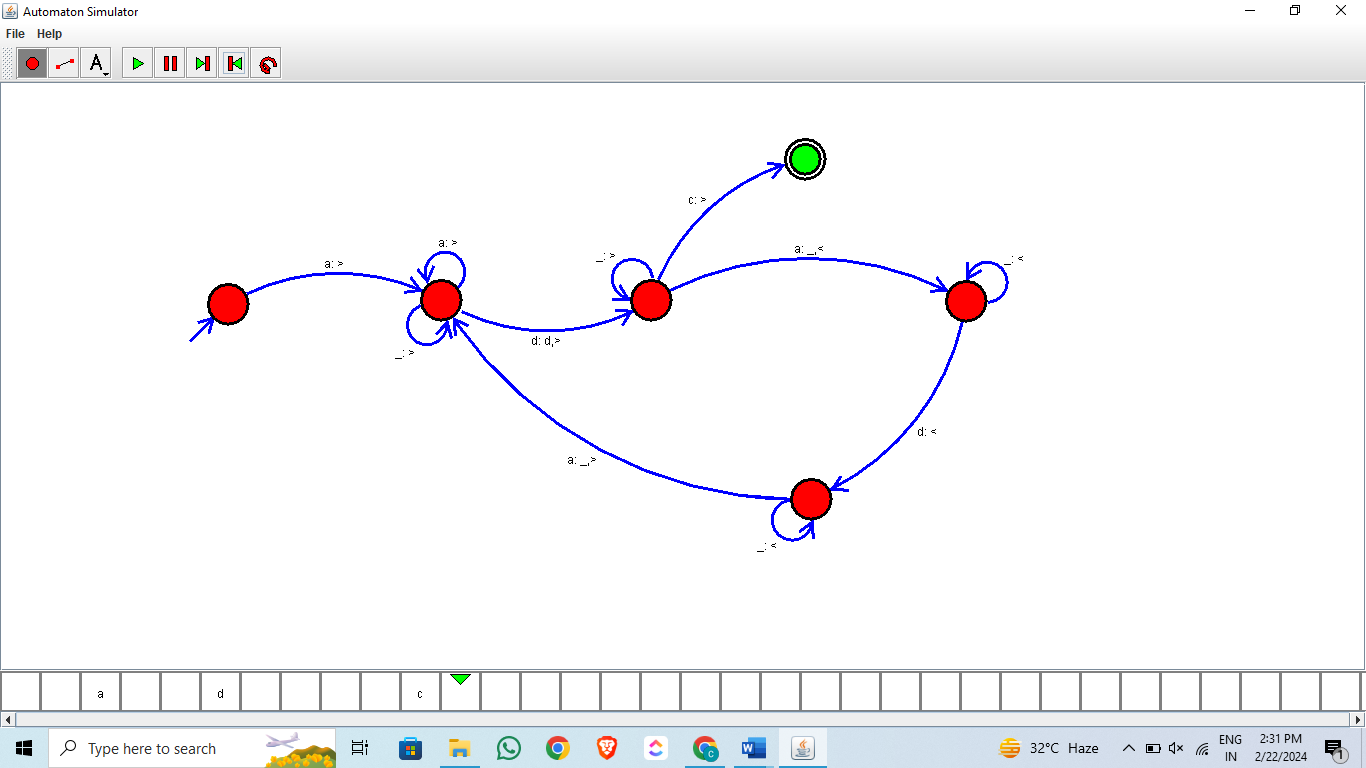
1. Design TM using simulator to accept the input string ww



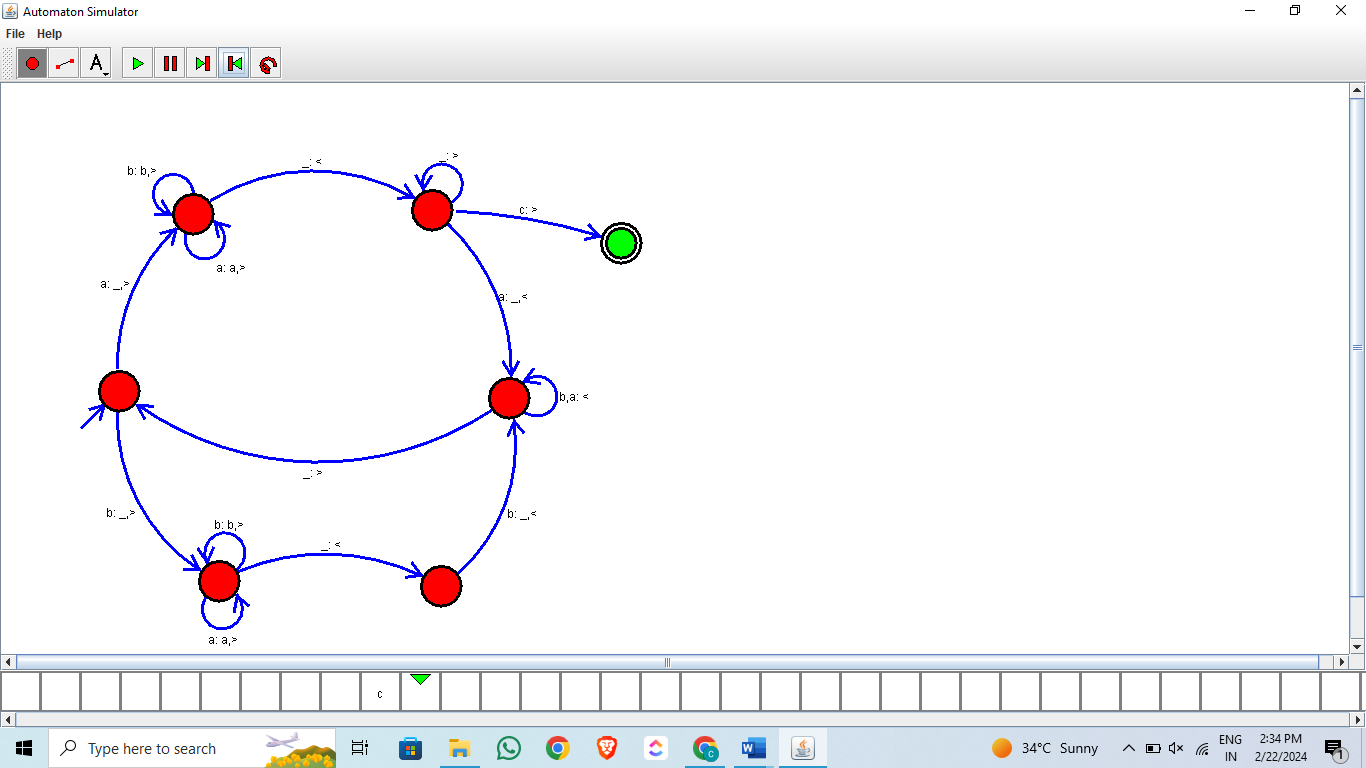
1. Design TM using simulator to perform addition of ‘aa’ and ‘aaa’



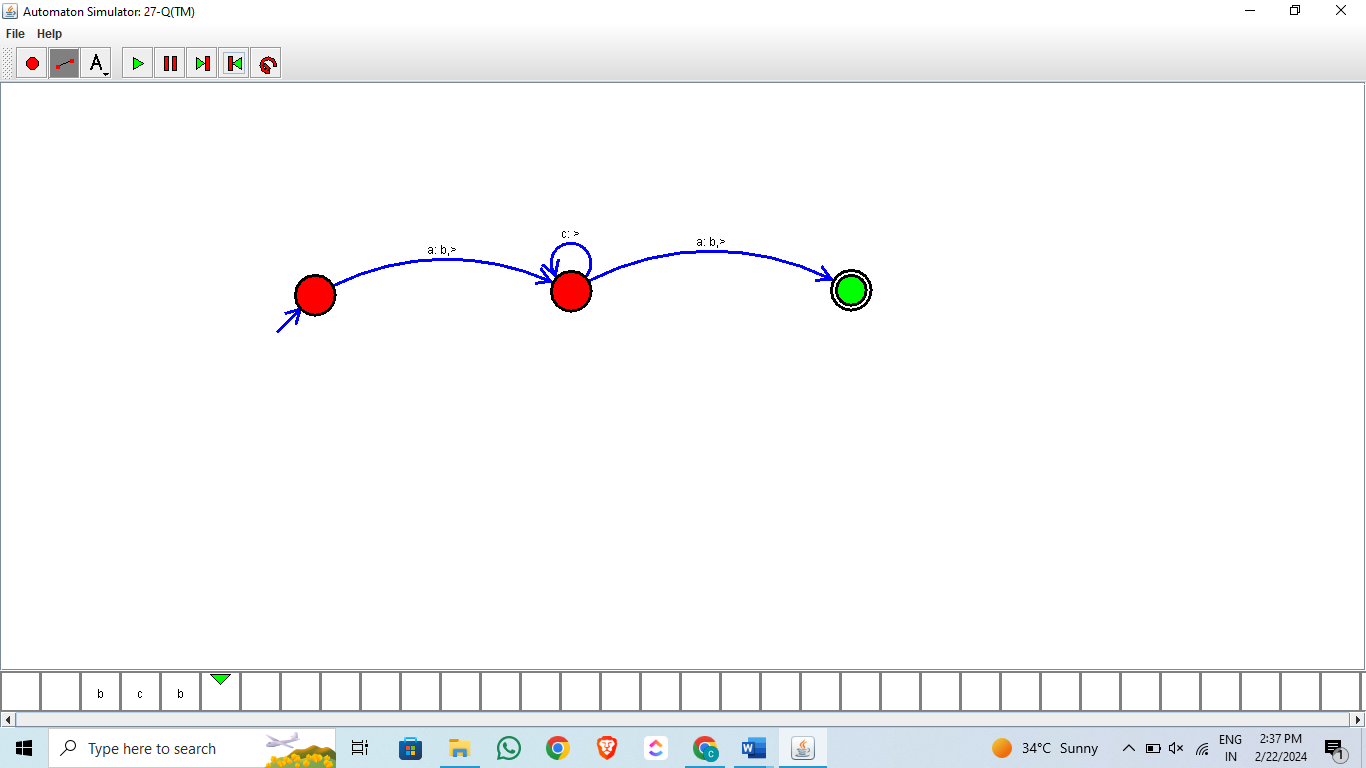
1. Design TM using simulator to perform subtraction of aaa-aa



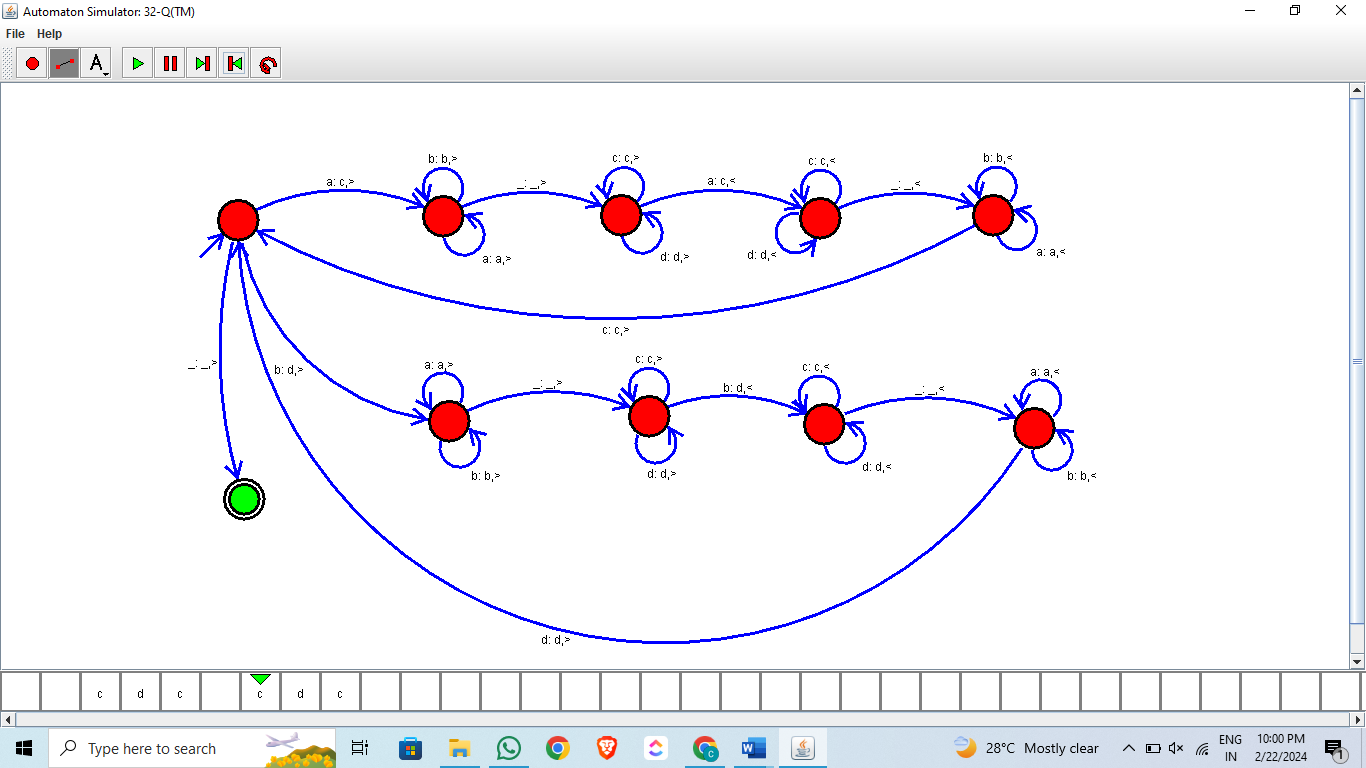
1. Design TM using simulator to accept the input string Palindrome bbabb



1. Design TM using simulator to accept the input string wcw



1. Design TM using simulator to perform string comparison where w={aba aba}



C-PROGRAM:

1. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with a and end with a

Code:

#include<stdio.h>

#include<string.h>

#define max 20

int main()

{

int trans\_table[4][2]={{1,3},{2,1},{2,1},{3,3}};

int final\_state=2,i;

int present\_state=0;

int next\_state=0;

int invalid=0;

char input\_string[max];

printf("Enter a string:");

scanf("%s",input\_string);

int l=strlen(input\_string);

for(i=0;i<l;i++)

{

if(input\_string[i]=='a')

next\_state=trans\_table[present\_state][0];

else if(input\_string[i]=='b')

next\_state=trans\_table[present\_state][1];

else

invalid=l;

present\_state=next\_state;

}

if(invalid==l)

{

printf("Invalid input");

}

else if(present\_state==final\_state)

printf("Accept\n");

else

printf("Don't Accept\n");

}

1. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with 0 and end with 1

Code:

#include<stdio.h>

#include<string.h>

#define max 20

int main()

{

int trans\_table[4][2]={{1,3},{1,2},{1,2},{3,3}};

int final\_state=2,i;

int present\_state=0;

int next\_state=0;

int invalid=0;

char input\_string[max];

printf("Enter a string:");

scanf("%s",input\_string);

int l=strlen(input\_string);

for(i=0;i<l;i++)

{

if(input\_string[i]=='0')

next\_state=trans\_table[present\_state][0];

else if(input\_string[i]=='1')

next\_state=trans\_table[present\_state][1];

else

invalid=l;

present\_state=next\_state;

}

if(invalid==l)

{

printf("Invalid input");

}

else if(present\_state==final\_state)

printf("Accept\n");

else

printf("Don't Accept\n");

}

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0A1 A → 0A | 1A | ε

Code:

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

for(i=0;i<l;i++)

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

if (s[0]=='0'&&s[l-1]=='1')

printf("string is accepted\n");

else

printf("string is Not accepted\n");

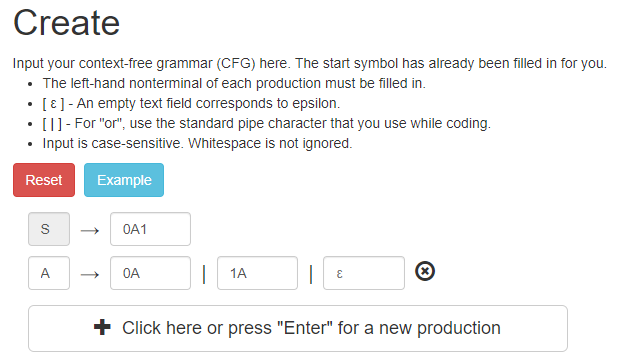
}

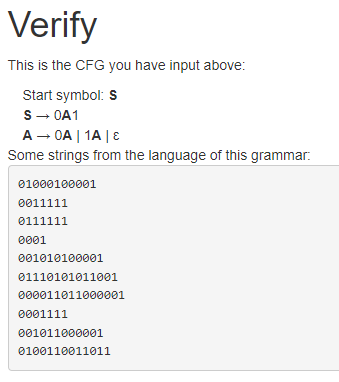
}

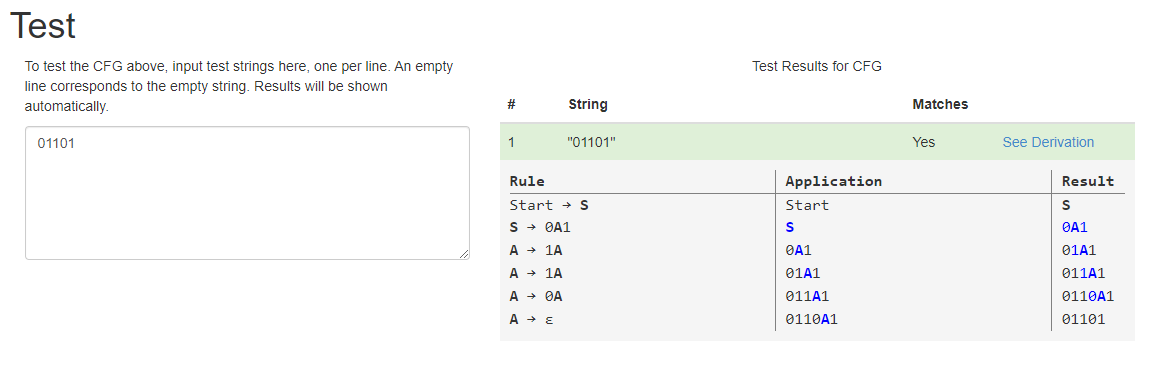
Output:

enter a string to check:0101

string is accepted







1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S0 | 1S1 | 0 | 1 | ε

Code:

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,a,b;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

for(i=0;i<l;i++)

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

flag1=1;

a=0;b=l-1;

while(a!=(l/2))

{

if(s[a]!=s[b])

{

flag1=0;

}

a=a+1;

b=b-1;

}

if (flag1==1)

{

printf("The string is a palindrome\n");

printf("string is accepted\n");

}

else

{

printf("The string is not a palindrome\n");

printf("string is Not accepted\n");

}

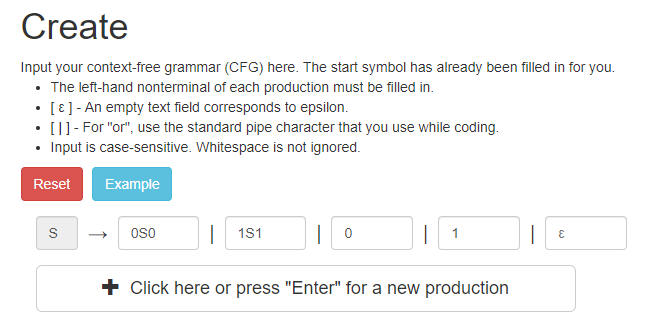
}

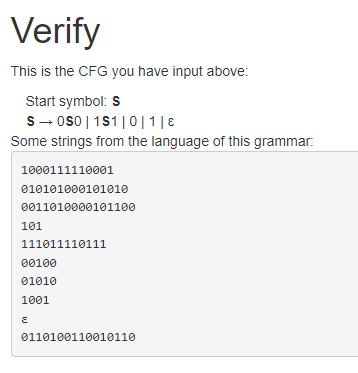
}

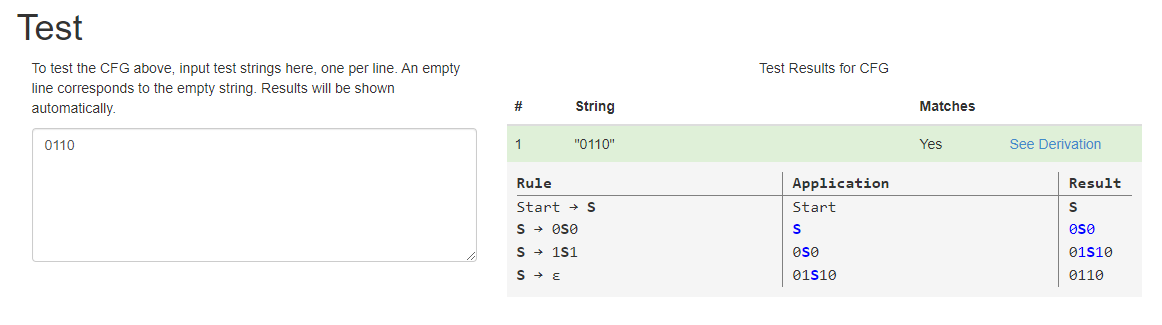
Ouput:

enter a string to check:0101

string is accepted







1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S0 | A A → 1A | ε

Code:

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,a,b;

int l,count1,count2;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

for(i=0;i<l;i++)

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

i=0;count1=0;

while(s[i]=='0')

{

count1++;

i++;

}

while(s[i]=='1')

{

i++;

}

flag1=1;

count2=0;

while(i<l)

{

if(s[i]=='0')

{

count2++;

}

else

{

flag1=0;

}

i++;

}

if(flag1==1)

{

if(count1==count2)

{

printf("The string satisfies the condition 0n1m0n\n");

printf("String Accepted\n");

}

else

{

printf("The string does not satisfy the condition 0n1m0n\n");

printf("String Not Accepted\n");

}

}

else

{

printf("The string does not satisfy the condition 0n1m0n\n");

printf("String Not Accepted\n");

}

}

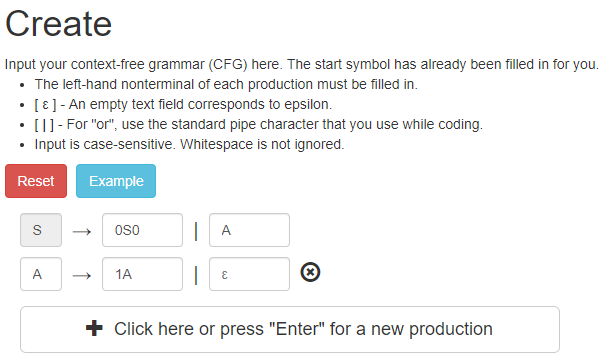
}

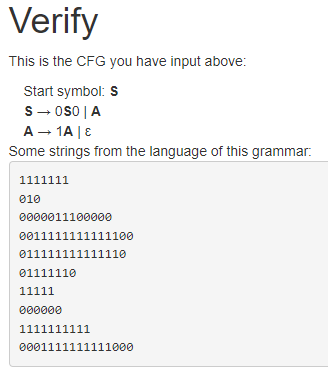
Output:

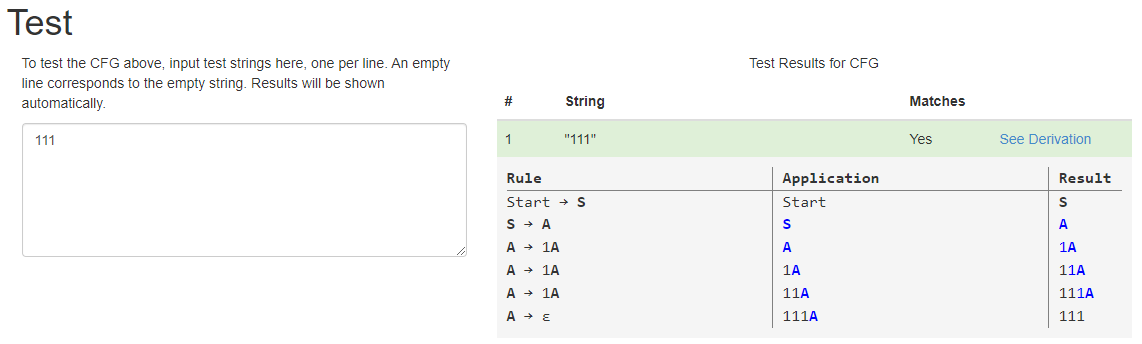
enter a string to check:111

The string satisfies the condition 0n1m0n

String Accepted







1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S1 | ε

Code:

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,flag2;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

for(i=0;i<l;i++)

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

if(l%2!=0)

{

printf("The string does not satisfy the condition 0n1n\n");

printf("String Not Accepted\n");

}

else

{

flag1=1;

for(i=0;i<(l/2);i++)

{

if(s[i]!='0')

{

flag1=0;

}

}

flag2=1;

for(i=l/2;i<l;i++)

{

if(s[i]!='1')

{

flag2=0;

}

}

if(flag1==1 && flag2==1)

{

printf("The string satisfies the condition 0n1n\n");

printf("String Accepted\n");

}

else

{

printf("The string does not satisfy the condition 0n1n\n");

printf("String Not Accepted\n");

}

}

}

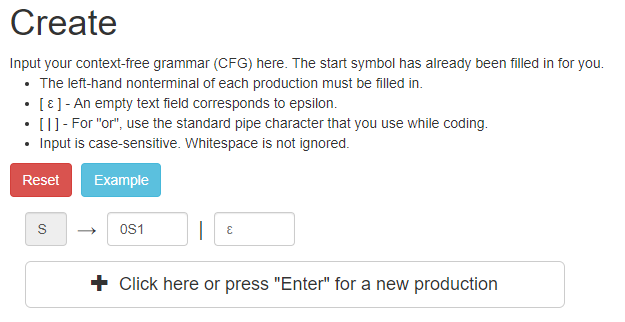
}

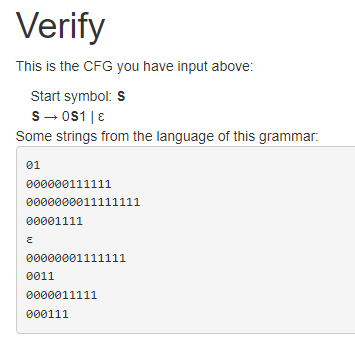
Output:

enter a string to check:000111

The string satisfies the condition 0n1n

String Accepted





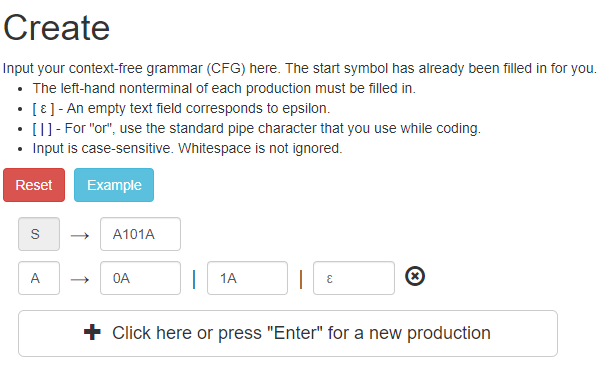


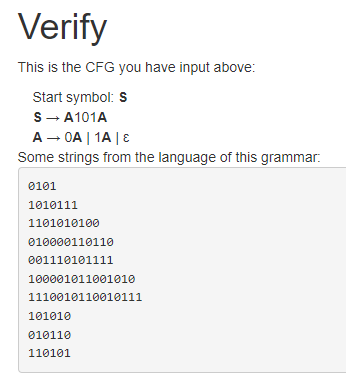
1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

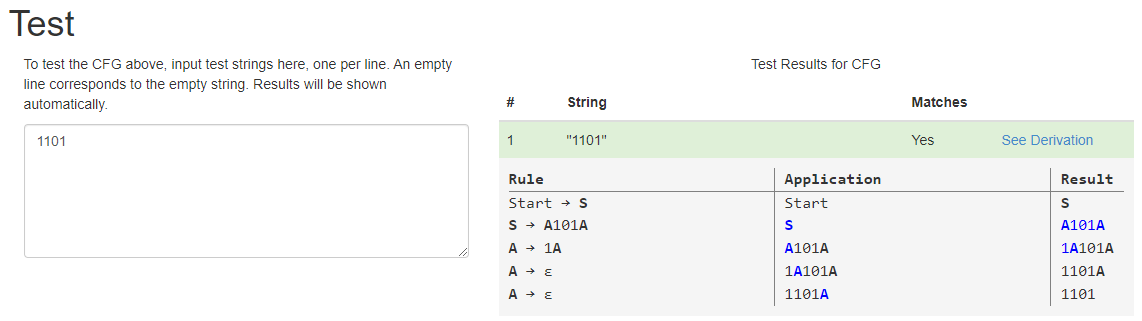
S → A101A, A → 0A | 1A | ε

Code:

Output:







1. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with b and end with a

Code:

#include <stdio.h>

#include <string.h>

#define MAX\_STRING\_LENGTH 100

int current\_state = 0;

int simulate\_NFA(char\* input\_string) {

int length = strlen(input\_string);

int i;

for (i = 0; i < length; i++) {

if (current\_state == 0 && input\_string[i] == 'b') {

current\_state = 1;

} else if (current\_state == 1 && input\_string[i] == 'a') {

current\_state = 2;

return 1;

} else {

current\_state = 0;

return 0;

}

}

return 0;

}

int main() {

char input\_string[MAX\_STRING\_LENGTH];

printf("Enter a string to check: ");

scanf("%s", input\_string);

if (simulate\_NFA(input\_string)) {

printf("Accepted\n");

} else {

printf("Rejected\n");

}

return 0;

}

Output:

Enter the string to check: baaba

Accepted

1. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given languagerepresenting strings that start with o and end with 1

Code:

#include <stdio.h>

#include <string.h>

#define MAX\_STRING\_LENGTH 100

int current\_state = 0;

int simulate\_NFA(char\* input\_string) {

int length = strlen(input\_string);

int i;

for (i = 0; i < length; i++) {

if (current\_state == 0 && input\_string[i] == '0') {

current\_state = 1;

} else if (current\_state == 1 && input\_string[i] == '1') {

current\_state = 2;

return 1;

} else {

current\_state = 0;

return 0;

}

}

return 0;

}

int main() {

char input\_string[MAX\_STRING\_LENGTH];

printf("Enter a string to check: ");

scanf("%s", input\_string);

if (simulate\_NFA(input\_string)) {

printf("Accepted\n");

} else {

printf("Rejected\n");

}

return 0;

}

Output:

Enter a string to check: 011101

Accepted

1. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

Code:

#include<stdio.h>

#include<string.h>

int trans\_table[10][5][3];

char symbol[5],a;

int e\_closure[10][10],ptr,state;

void find\_e\_closure(int x);

int main()

{

int i,j,k,n,num\_states,num\_symbols;

for(i=0;i<10;i++)

{

for(j=0;j<5;j++)

{

for(k=0;k<3;k++)

{

trans\_table[i][j][k]=-1;

}

}

}

printf("How may states in the NFA with e-moves:");

scanf("%d",&num\_states);

printf("How many symbols in the input alphabet including e :");

scanf("%d",&num\_symbols);

printf("Enter the symbols without space. Give 'e' first:");

scanf("%s",symbol);

for(i=0;i<num\_states;i++)

{

for(j=0;j<num\_symbols;j++)

{

printf("How many transitions from state %d for the input %c:",i,symbol[j]);

scanf("%d",&n);

for(k=0;k<n;k++)

{

printf("Enter the transitions %d from state %d for the input %c :", k+1,i,symbol[j]);

scanf("%d",&trans\_table[i][j][k]);

}

}

}

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

e\_closure[i][j]=-1;

}

}

for(i=0;i<num\_states;i++)

e\_closure[i][0]=i;

for(i=0;i<num\_states;i++)

{

if(trans\_table[i][0][0]==-1)

continue;

else

{

state=i;

ptr=1;

find\_e\_closure(i);

}

}

for(i=0;i<num\_states;i++)

{

printf("e-closure(%d)= {",i);

for(j=0;j<num\_states;j++)

{

if(e\_closure[i][j]!=-1)

{

printf("%d, ",e\_closure[i][j]);

}

}

printf("}\n");

}

}

void find\_e\_closure(int x)

{

int i,j,y[10],num\_trans;

i=0;

while(trans\_table[x][0][i]!=-1)

{

y[i]=trans\_table[x][0][i];

i=i+1;

}

num\_trans=i;

for(j=0;j<num\_trans;j++)

{

e\_closure[state][ptr]=y[j];

ptr++;

find\_e\_closure(y[j]);

}

}

Output:

How may states in the NFA with e-moves: 3

How many symbols in the input alphabet including e: 3

Enter the symbols without space. Give 'e' first: eab

How many transitions from state 0 for the input e: 2

Enter the transitions 1 from state 0 for the input e: 1

Enter the transitions 2 from state 0 for the input e: 2

How many transitions from state 0 for the input a: 1

Enter the transitions 1 from state 0 for the input a: 1

How many transitions from state 0 for the input b: 0

How many transitions from state 1 for the input e: 0

How many transitions from state 1 for the input a: 1

Enter the transitions 1 from state 1 for the input a: 2

How many transitions from state 1 for the input b: 0

How many transitions from state 2 for the input e: 0

How many transitions from state 2 for the input a: 0

How many transitions from state 2 for the input b: 1

Enter the transitions 1 from state 2 for the input b: 0

e-closure(0)= {0, 1, 2, }

e-closure(1)= {1, 2, }

e-closure(2)= {2, }

1. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

Code:

#include<stdio.h>

#include<string.h>

int trans\_table[10][5][3];

char symbol[5],a;

int e\_closure[10][10],ptr,state;

void find\_e\_closure(int x);

int main()

{

int i,j,k,n,num\_states,num\_symbols;

for(i=0;i<10;i++)

{

for(j=0;j<5;j++)

{

for(k=0;k<3;k++)

{

trans\_table[i][j][k]=-1;

}

}

}

printf("How may states in the NFA with e-moves:");

scanf("%d",&num\_states);

printf("How many symbols in the input alphabet including e :");

scanf("%d",&num\_symbols);

printf("Enter the symbols without space. Give 'e' first:");

scanf("%s",symbol);

for(i=0;i<num\_states;i++)

{

for(j=0;j<num\_symbols;j++)

{

printf("How many transitions from state %d for the input %c:",i,symbol[j]);

scanf("%d",&n);

for(k=0;k<n;k++)

{

printf("Enter the transitions %d from state %d for the input %c :", k+1,i,symbol[j]);

scanf("%d",&trans\_table[i][j][k]);

}

}

}

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

e\_closure[i][j]=-1;

}

}

for(i=0;i<num\_states;i++)

e\_closure[i][0]=i;

for(i=0;i<num\_states;i++)

{

if(trans\_table[i][0][0]==-1)

continue;

else

{

state=i;

ptr=1;

find\_e\_closure(i);

}

}

for(i=0;i<num\_states;i++)

{

printf("e-closure(%d)= {",i);

for(j=0;j<num\_states;j++)

{

if(e\_closure[i][j]!=-1)

{

printf("%d, ",e\_closure[i][j]);

}

}

printf("}\n");

}

}

void find\_e\_closure(int x)

{

int i,j,y[10],num\_trans;

i=0;

while(trans\_table[x][0][i]!=-1)

{

y[i]=trans\_table[x][0][i];

i=i+1;

}

num\_trans=i;

for(j=0;j<num\_trans;j++)

{

e\_closure[state][ptr]=y[j];

ptr++;

find\_e\_closure(y[j]);

}

}

Output:

How may states in the NFA with e-moves: 3

How many symbols in the input alphabet including e: 2

Enter the symbols without space. Give 'e' first: ea

How many transitions from state 0 for the input e: 1

Enter the transitions 1 from state 0 for the input e: 1

How many transitions from state 0 for the input a: 1

Enter the transitions 1 from state 0 for the input a: 2

How many transitions from state 1 for the input e: 0

How many transitions from state 1 for the input a: 1

Enter the transitions 1 from state 1 for the input a: 0

How many transitions from state 2 for the input e: 1

Enter the transitions 1 from state 2 for the input e: 1

How many transitions from state 2 for the input a: 0

e-closure(0)= {0, 1, 2, }

e-closure(1)= {1, 2, }

e-closure(2)= {1, 2, }