

**NAME: N KARTHIK REDDY**

**Reg.no.:192111167**

## **1.DETERMINISTIC FINITE AUTOMATA**

### **INPUT:**

```
#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]=='a')
            next_state=trans_table[present_state][0];
        else if(input_string[i]=='b')
            next_state=trans_table[present_state][1];
        else
            invalid=1;
        present_state=next_state;
    }
    if(invalid==1)
    {
```

```

        printf("invalid input");

    }

    else if(present_state==final_state)
        printf("accept\n");
    else
        printf("dont accept\n");
}

```

## OUTPUT:

```

C:\Users\Luckysushu\Desktop\TOC\DFA.exe
enter a string: abaab
accept

-----
Process exited after 2.877 seconds with return value 0
Press any key to continue . . .

```

## 2.NON-DETERMINISTIC FINITE AUTOMATA:

### INPUT:

```

#include<stdio.h>

#include<string.h>

int main()
{

```

```

int i,j,k,l,m,next_state[20],n,mat[10][10][10],flag,p,exit;
int num_states,final_state[5],num_symbols,num_final;
int present_state[20],prev_trans,new_trans;
char ch,input[20];
int symbol[5],inp,inp1;
printf("how many states in the nfa: ");
scanf("%d",&num_states);
printf("how many symbols in the input alphabet: ");
scanf("%d",&num_symbols);
for(i=0;i<num_symbols;i++)
{
    printf("enter the input symbol %d: ",i+1);
    scanf("%d",&symbol[i]);
}
printf("how many final states: ");
scanf("%d",&num_final);
for(i=0;i<num_final;i++)
{
    printf("enter the final state %d: ",i+1);
    scanf("%d",&final_state[i]);
}
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            mat[i][j][k]=-1;
        }
    }
}

```

```

    }
}
for(i=0;i<num_states;i++)
{
    for(j=0;j<num_symbols;j++)
    {
        printf("how many transitions from state %d for the input %d:
",i,symbol[j]);

        scanf("%d",&n);
        for(k=0;k<n;k++)
        {
            printf("enter the transition %d from state %d for the input %d:
",k+1,i,symbol[j]);

            scanf("%d",&mat[i][j][k]);

        }
    }
}

printf("the transitions are stored as below\n");
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            if(mat[i][j][k]!=-1)
                printf("mat[%d][%d][%d]=%d\n",i,j,k,mat[i][j][k]);

        }
    }
}

while(1)

```

```

{
    printf("enter the input string: ");
    scanf("%s",input);
    present_state[0]=0;
    prev_trans=1;
    l=strlen(input);
    for(i=0;i<l;i++)
    {
        if(input[i]=='0')
            inp1=0;
        else if(input[i]=='1')
            inp=1;
        else
        {
            printf("invalid input\n");
            exit;
        }
        for(m=0;m<num_symbols;m++)
        {
            if(inp1==symbol[m])
            {
                inp=m;
                break;
            }
        }
        new_trans=0;
        for(j=0;j<prev_trans;j++)
        {
            k=0;
            p=present_state[j];

```

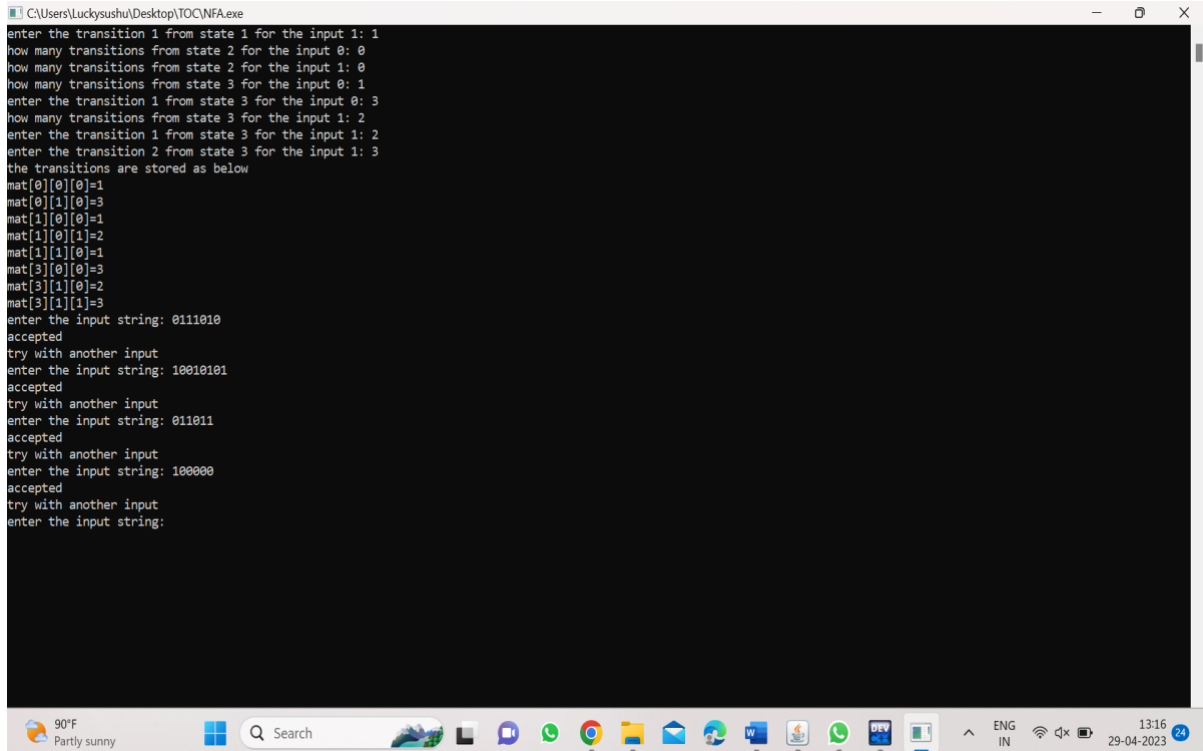
```

        while(mat[p][inp][k]!=-1)
        {
            next_state[new_trans++]=mat[p][inp][k];
            k++;
        }
    }
    for(j=0;j<new_trans;j++)
    {
        present_state[j]=next_state[j];
    }
    prev_trans=new_trans;
}
flag=0;
for(i=0;i<prev_trans;i++)
{
    for(j=0;j<num_final;j++)
    {
        if(present_state[i]==final_state[j])
        {
            flag=1;
            break;
        }
    }
}
if(flag==1)
printf("accepted\n");
else
printf("not accepted\n");
printf("try with another input\n");
}

```

```
}
```

## OUTPUT:



```
C:\Users\Luckyshu\Desktop\TOC\NFA.exe
enter the transition 1 from state 1 for the input 1: 1
how many transitions from state 2 for the input 0: 0
how many transitions from state 2 for the input 1: 0
how many transitions from state 3 for the input 0: 1
enter the transition 1 from state 3 for the input 0: 3
how many transitions from state 3 for the input 1: 2
enter the transition 1 from state 3 for the input 1: 2
enter the transition 2 from state 3 for the input 1: 3
the transitions are stored as below
mat[0][0][0]=1
mat[0][1][0]=3
mat[1][0][0]=1
mat[1][0][1]=2
mat[1][1][0]=1
mat[3][0][0]=3
mat[3][1][0]=2
mat[3][1][1]=3
enter the input string: 0111010
accepted
try with another input
enter the input string: 10010101
accepted
try with another input
enter the input string: 011011
accepted
try with another input
enter the input string: 100000
accepted
try with another input
enter the input string:
```

## 3.EPSILON CLOSURE FOR NFA

### INPUT:

```
#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 many 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:",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");

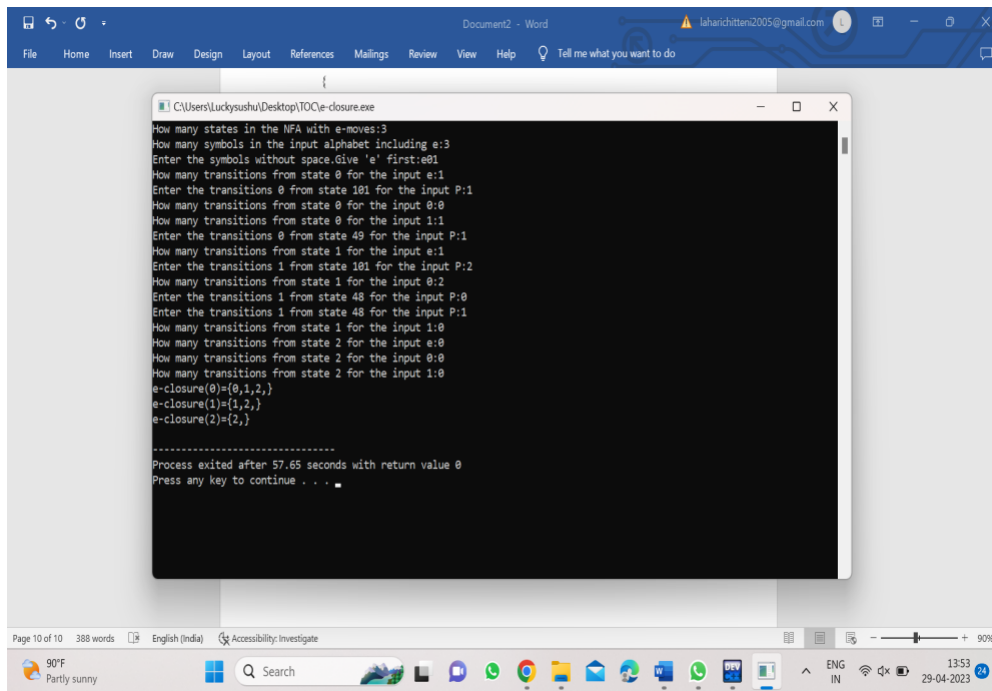
```

```

}
}
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:**



## 4.CHECKING STRING BELONGS TO THE GRAMMAR

### INPUT:

```
#include<stdio.h>
```

```
#include<string.h>
```

```
int main()
```

```
{
```

```
    char s[100];
```

```
    int i,flag;
```

```
    int l;
```

```
    printf("enter the 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");
}
}

```

## OUTPUT:

The screenshot shows a Dev-C++ window with the file '4a-string.cpp'. The code is as follows:

```

1  #include<stdio.h>
2
3  int main()
4  {
5      char s[100];
6      int l, flag=1;
7      printf("Enter the string to check:\n");
8      scanf("%s", s);
9      l=strlen(s);
10     flag=1;
11     for(i=0; i<l; i++)
12     {
13         if(s[i]!='0' && s[i]!='1')
14             flag=0;
15     }
16     if(flag!=1)
17         printf("string is not valid\n");
18     if(flag==1)
19     {
20         if(s[0]=='0' && s[l-1]=='1')
21             printf("string is accepted\n");
22         else
23             printf("string is not accepted\n");
24     }
25 }

```

The console output shows: "Enter the string to check: 01011011101" followed by "string is accepted". The status bar at the bottom indicates "Done parsing in 0.031 seconds".

## 5.CHECKING WHETHER STRING BELONGS TO THE GRAMMAR

### INPUT:

```
#include<stdio.h>
```

```

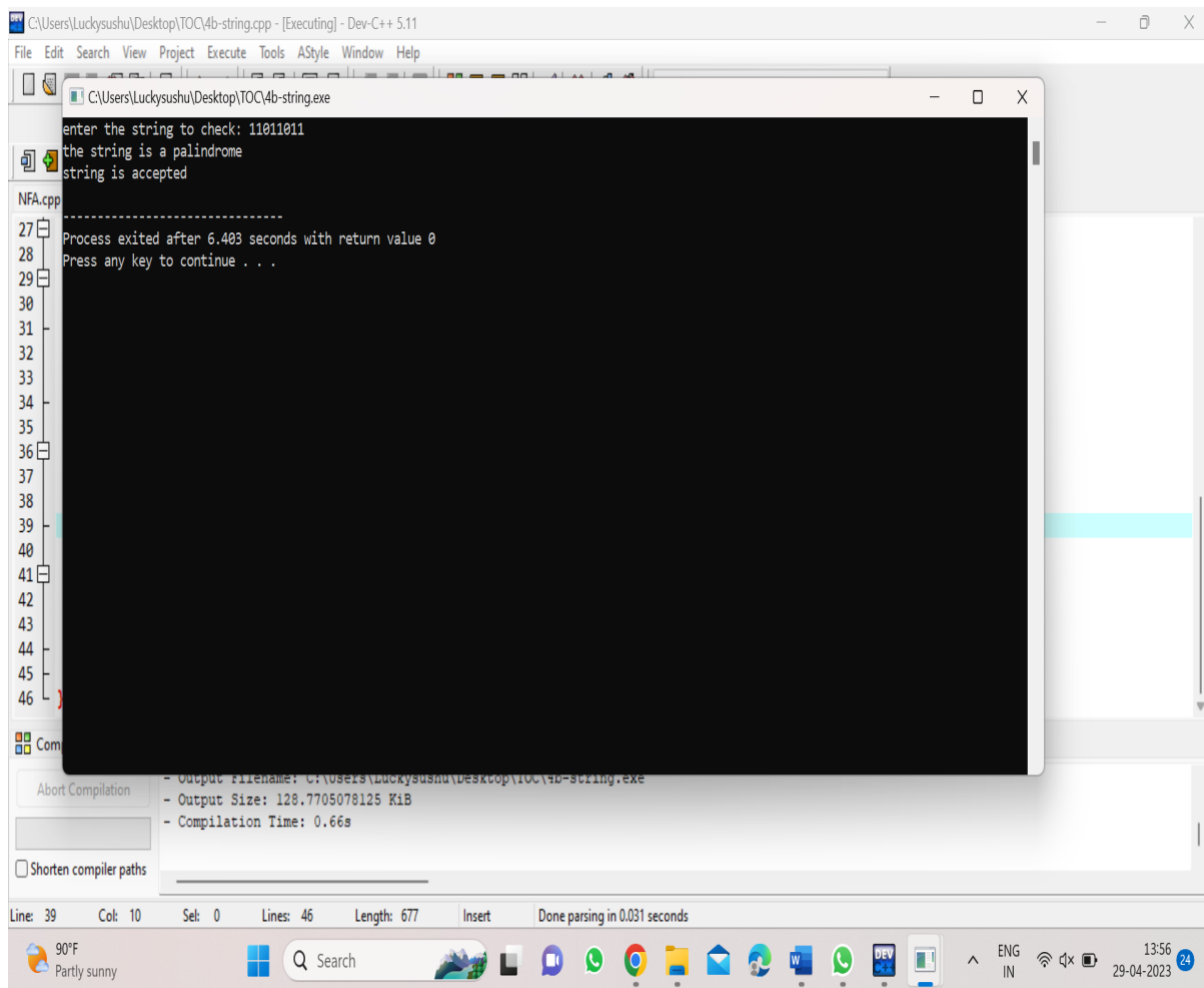
#include<string.h>

int main()
{
    char s[100];
    int i,flag,flag1,a,b;
    int l;
    printf("enter the 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");
    }
}
}
```

**OUTPUT:**



## 6.CHECKING WHETHER STRING BELONGS TO THE GRAMMAR

### INPUT:

```
#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++;
        }
    }
}
```

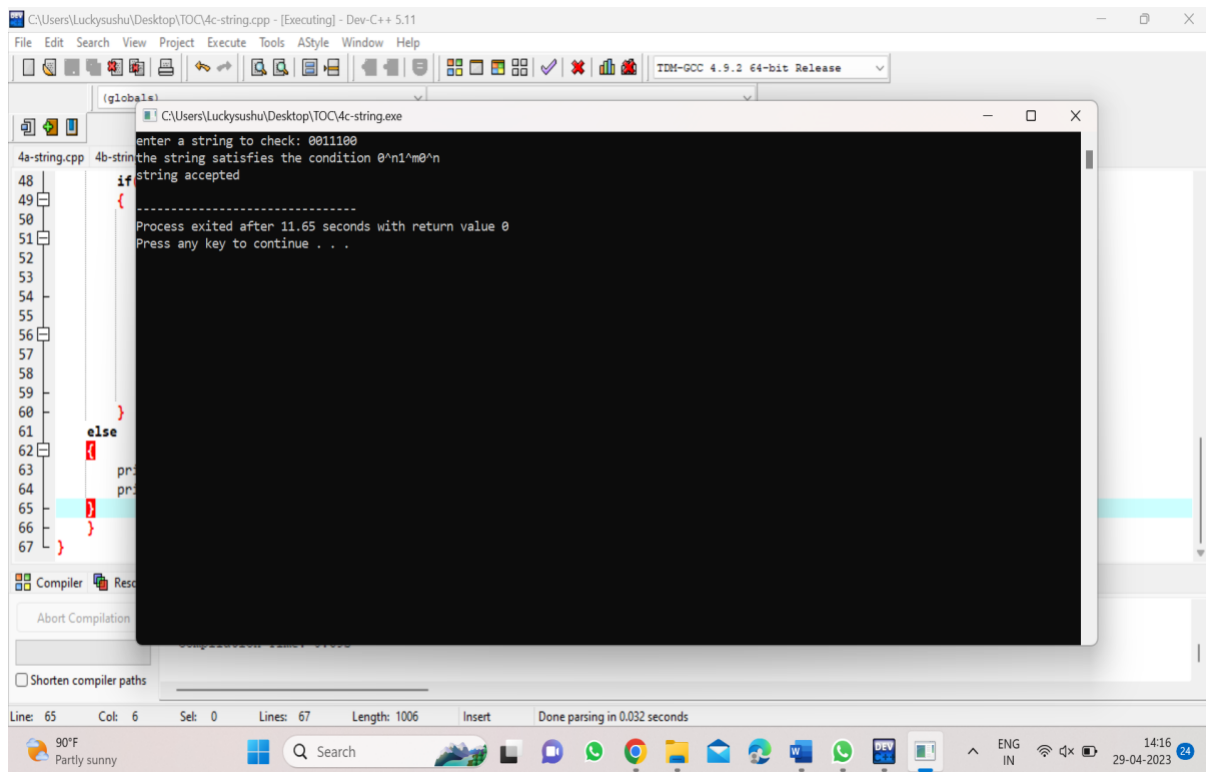


```

        }
        else
        {
            flag1=0;
        }
        i++;
    }
    if(flag1==1)
    {
        if(count1==count2)
        {
            printf("the string satisfies the condition 0^n1^m0^n\n");
            printf("string accepted\n");
        }
        else
        {
            printf("the string does not satisfy the condition 0^n1^m0^n\n");
            printf("string not accepted\n");
        }
    }
    else
    {
        printf("the string does not satisfy the condition 0^n1^m0^n\n");
        printf("string not accepted\n");
    }
}

```

**OUTPUT:**



## 7.CHECKING WHETHER STRING BELONGS TO THE GRAMMAR

### INPUT:

```
#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 doesnot satisfy the condition 0^n1^n\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 0^n1^n\n");
        printf("string accepted\n");

    }
    else
    {

        printf("the string does not satisfies the condition 0^n1^n\n");
        printf("string not accepted\n");

    }

}

}
}

```

## OUTPUT:

The screenshot shows a Dev-C++ IDE window titled "C:\Users\Luckyshu\Desktop\TOCAd-string.cpp - [Executing] - Dev-C++ 5.11". The main window displays the execution of a C program. The program prompts the user to "Enter a string to check: 0000011111". The output shows "the string satisfies the condition 0^n1^n" and "string accepted". The compiler window shows the output file name "C:\Users\Luckyshu\Desktop\TOCAd-string.exe", output size "128.7705078125 KiB", and compilation time "0.63s".

## 8.CHECKING WHETHER STRING BELONGS TO THE GRAMMAR

### INPUT:

```

#include<stdio.h>
#include<string.h>
int main()
{
    char s[100];
    int i,flag,flag1;
    int l;
    printf("enter the 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 valid\n");
    else
        printf("string is not valid\n");
    if(flag==1)
    {
        flag1=0;
        for(i=0;i<l-2;i++)
        {
            if(s[i]=='1')
            {
                if(s[i+1]=='0' && s[i+2]=='1')

```

```

        {
            flag1=1;
            printf("substring 101 exists,string is accepted\n");
            break;
        }
    }
}
if(flag1==0)
    printf("substring 101 does not exist.string is not accepted\n");
}
}

```

## OUTPUT:

The screenshot shows the Dev-C++ IDE with the following details:

- File Explorer:** Shows the file `4a-string.cpp` at line 41.
- Compiler Output:**
  - Output Filename: `C:\Users\Luckysushu\Desktop\TOC\4e-string.exe`
  - Output Size: `128.7705078125 KiB`
  - Compilation Time: `0.64s`
- Console Window:**
  - Enter the string to check: `100100101110`
  - string is valid
  - substring 101 exists,string is accepted
  - Process exited after 7.376 seconds with return value 0
  - Press any key to continue . . .
- Status Bar:** Line: 36, Col: 14, Sel: 0, Lines: 41, Length: 679, Insert, Done parsing in 0.047 seconds.
- Taskbar:** Shows the system clock as 14:41 on 29-04-2023, with weather (90°F, Partly sunny) and various application icons.

## 9.SIMULATING PUSHDOWN AUTOMATA ( $0^n1^n$ ):

### INPUT:

```
#include<stdio.h>
```

```

#include<string.h>

char stack[20];

int top;

int push()
{
    top=top+1;
    stack[top]='0';
    stack[top+1]='\0';
}

int pop()
{
    if(top<1)
        return(0);
    else
    {
        stack[top]='\0';
        top=top-1;
        return(1);
    }
}

int main()
{
    int m,i,j,k,l,a,len;
    char input[20],rem_input[20];
    printf("Simulation of Pushdown Automata for 0n1n\n");
    printf("Enter a string : ");
    scanf("%s",input);
    l=strlen(input);
    j=0;stack[0]='Z';top=0;
    printf("Stack\tInput\n");

```

```

        printf("%s\t%s\n",stack,input);
while(1)
{
    len=strlen(input);
    while(len>0)
    {
        if(input[0]=='0')
        {
            push();

            m=0;
            for(k=1;k<len;k++)
            {
                rem_input[m]=input[k];
                m=m+1;
            }
            rem_input[m]='\0';

            strcpy(input,rem_input);
            printf("%s\t%s\n",stack,input);
        }
        if(input[0]=='1')
        {
            a=pop();
            if(a==0)
            {
                printf("String not accepted");

                goto b;
            }
            else
            {
                m=0;

```



```

        for(k=1;k<len;k++)
        {
            rem_input[m]=input[k];
            m=m+1;
        }
        rem_input[m]='\0';

                                strcpy(input,rem_input);

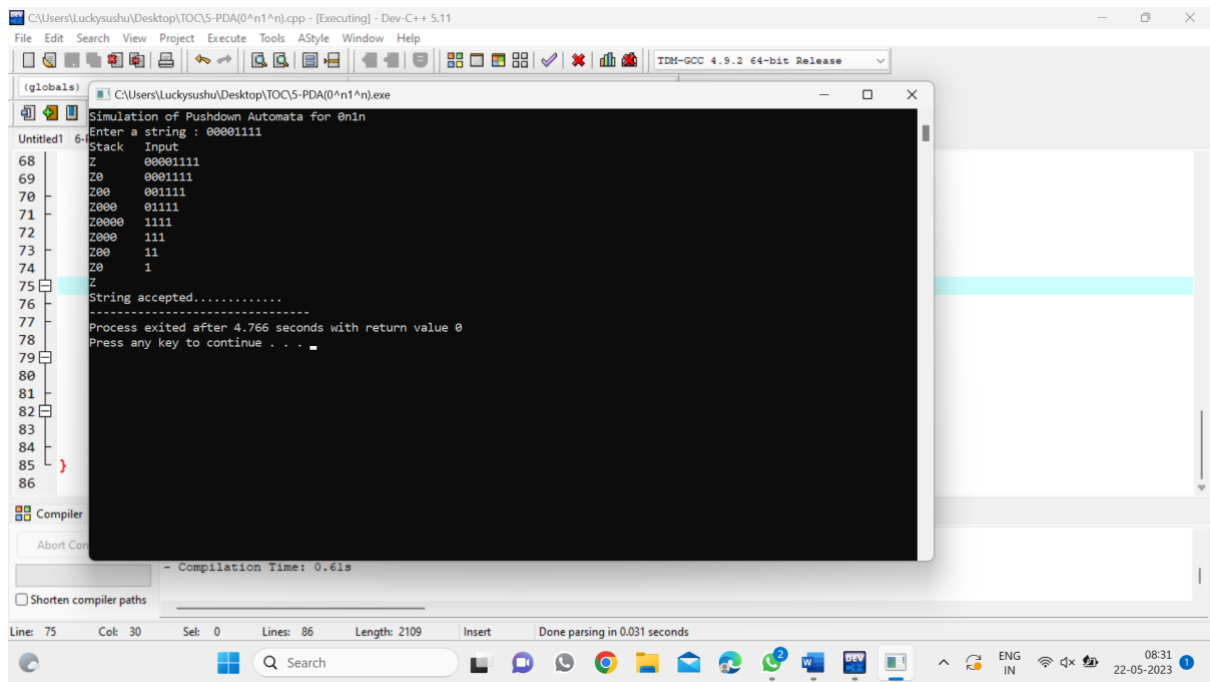
        printf("%s\t%s\n",stack,input);
    }
}

break;
}    j=j+1;
if(j==(1))
{    break;
}

}
if(top>=1)
{
    printf("String not accepted");
} else
{
    printf("String accepted");
}    b:    printf(".....");
}

```

**OUTPUT:**



## 10)SIMULATING PUSHDOWN AUTOMATA ( $0^n1^{2n}$ ):

### INPUT:

```
#include<stdio.h>
```

```
#include<string.h>
```

```
char stack[20];
```

```
int top,count=0;
```

```
int push()
```

```
{
```

```
    top=top+1;
```

```
    stack[top]='0';
```

```
    stack[top+1]='\0';
```

```
}
```

```
int pop()
```

```
{
```

```
if(top<1)
```

```
return(0);
```

```

else
{
    stack[top]='\0';
        top=top-1;
    return(1);
}
}

int main()
{
    int m,i,j,k,l,a,len;
    char input[20],rem_input[20];
    printf("Simulation of PDA for n 0's followed by 2n 1's\n");
    printf("Enter a string : ");
        scanf("%s",input);
        l=strlen(input);
        j=0;
        stack[0]='Z';
        top=0;
        printf("Stack\tInput\n");
        printf("%s\t%s\n",stack,input);
    while(1)
    {
        len=strlen(input);
        while(len>0)
        {
            if(input[0]=='0')
            {
                push();

                m=0;
                for(k=1;k<len;k++)

```

```

{
    rem_input[m]=input[k];
    m=m+1;
}
rem_input[m]='\0';
strcpy(input,rem_input);
printf("%s\t%s\n",stack,input);
}
if(input[0]=='1')
{
    count++;

    if(count%2==0)
    {
        a=pop();
        if(a==0)
        {
            printf("String not accepted");

            goto b;
        }
        else
        {
            m=0;

            for(k=1;k<len;k++)
            {
                rem_input[m]=input[k];
                m=m+1;
            }
        }
        rem_input[m]='\0';

        strcpy(input,rem_input);
    }
}

```

```

        printf("%s\t%s\n",stack,input);
    }        else        {
        m=0;

                for(k=1;k<len;k++)

        {
            rem_input[m]=input[k];
            m=m+1;
        }
        rem_input[m]='\0';

                strcpy(input,rem_input);

        printf("%s\t%s\n",stack,input);
    }
}

break;
}

j=j+1;


        if(j==1)
        {
            break;
        }
}

if(top>=1)
{
    printf("String not accepted");
} else
{
    printf("String accepted");
} b:
    printf(".....");
}

```

The screenshot displays a Windows desktop environment. The primary application is Dev-C++ 5.11, which is running a C++ program. The program's output, visible in the console window, is as follows:

```

Simulation of PDA for n 0's followed by 2n 1's
Enter a string : 000111111
Stack   Input
Z       000111111
Z0      00111111
Z00     01111111
Z000    11111111
Z000    111111
Z000    1111
Z000    111
Z000    11
Z000    1
Z
String accepted.....
-----
if (
{ Process exited after 5.639 seconds with return value 0
  Press any key to continue . . .
}
}

```

The IDE window title is "C:\Users\LuckySushu\Desktop\TOC\6-PDA.cpp - [Executing] - Dev-C++ 5.11". The taskbar at the bottom shows the system clock as 08:54 on 22-05-2023, along with various application icons.

```
flag2=0;i=0;
```

```
while(i<le)
{
    if((str[i]=='0')&&(flag==0))
    {
        str[i] = 'A';
        printf("%s\n",str);
        flag=1;
        i=i+1;
    }
    else if((str[i]=='0')&&(flag==1))
    {
        i=i+1;
    }
    else if(str[i]=='A')
    {
        i=i+1;
    }
    else if((str[i]=='1')&&(flag1==0))
    {
        str[i] = 'B';
        printf("%s\n",str);
        flag1=1;
        i=i+1;
    }
    else if((str[i]=='1')&&(flag1==1))
    {
        i=i+1;
    }
    else if(str[i]=='B')
    {
```

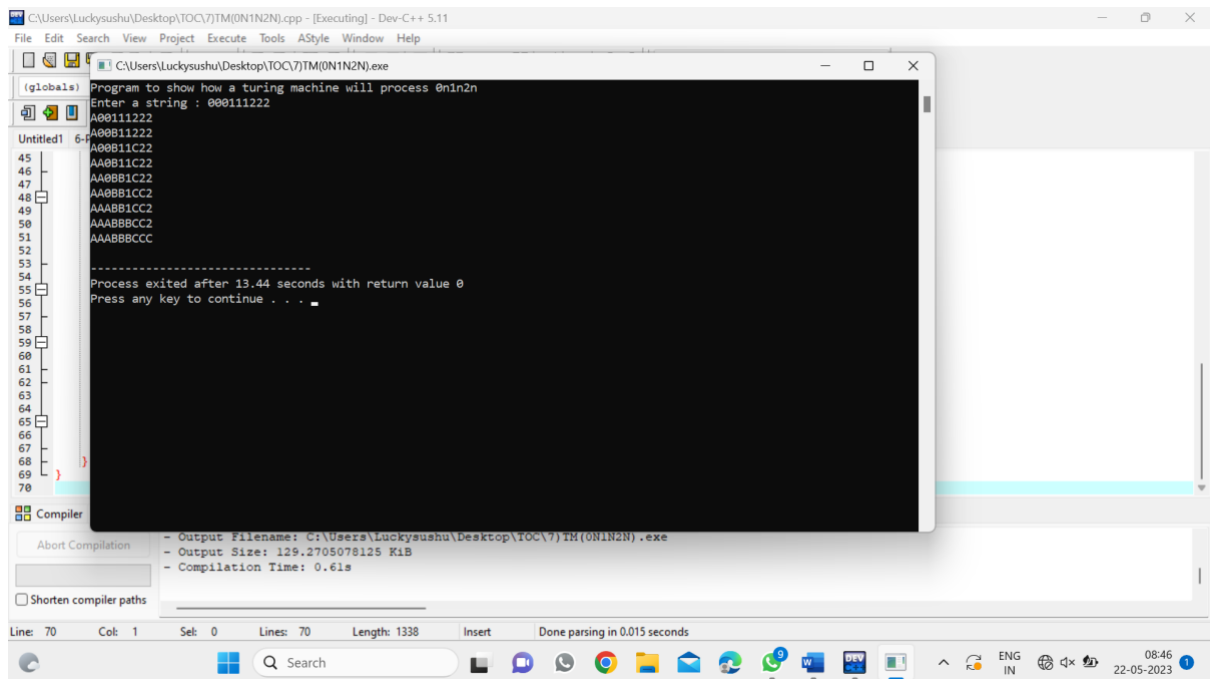
```

        i=i+1;
    }
    else if((str[i]=='2')&&(flag2==0))
    {
        str[i]='C';
        printf("%s\n",str);
        flag2=1;
        i=i+1;
    }
    else if((str[i]=='2')&&(flag2==1))
    {
        i=i+1;
    }
    else if(str[i]=='C')
    {
        i=i+1;
    }
}
j=j+1;
if(j==le)
{
    break;
}
}
}

```



## OUTPUT:



```
C:\Users\Luckysushu\Desktop\TOC\7\TM(0N1N2N).cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

C:\Users\Luckysushu\Desktop\TOC\7\TM(0N1N2N).exe
(globals) Program to show how a turing machine will process 0n1n2n
Enter a string : 000111222
AG0111222
AA00B11222
AA00B11C22
AA00B1C22
AA00B1CC2
AA00B1CC2
AA00BBCC2
AA00BBCCC
-----
Process exited after 13.44 seconds with return value 0
Press any key to continue . . .
```

Compiler

- Output Filename: C:\Users\Luckysushu\Desktop\TOC\7\TM(0N1N2N).exe
- Output Size: 129.2705078125 KiB
- Compilation Time: 0.61s

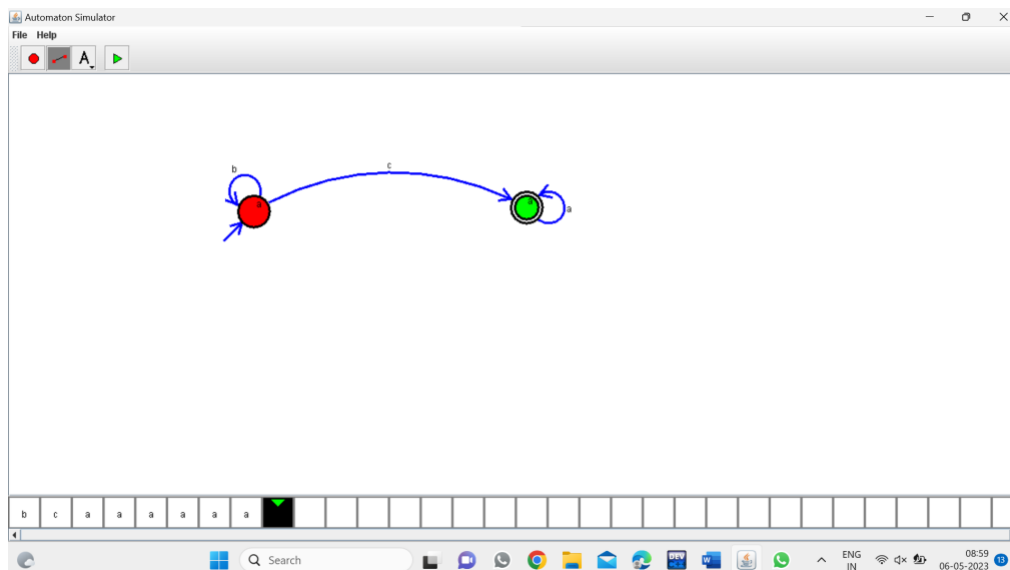
Shorten compiler paths

Line: 70 Col: 1 Sel: 0 Lines: 70 Length: 1338 Insert Done parsing in 0.015 seconds

## SIMULATION PROGRAMS

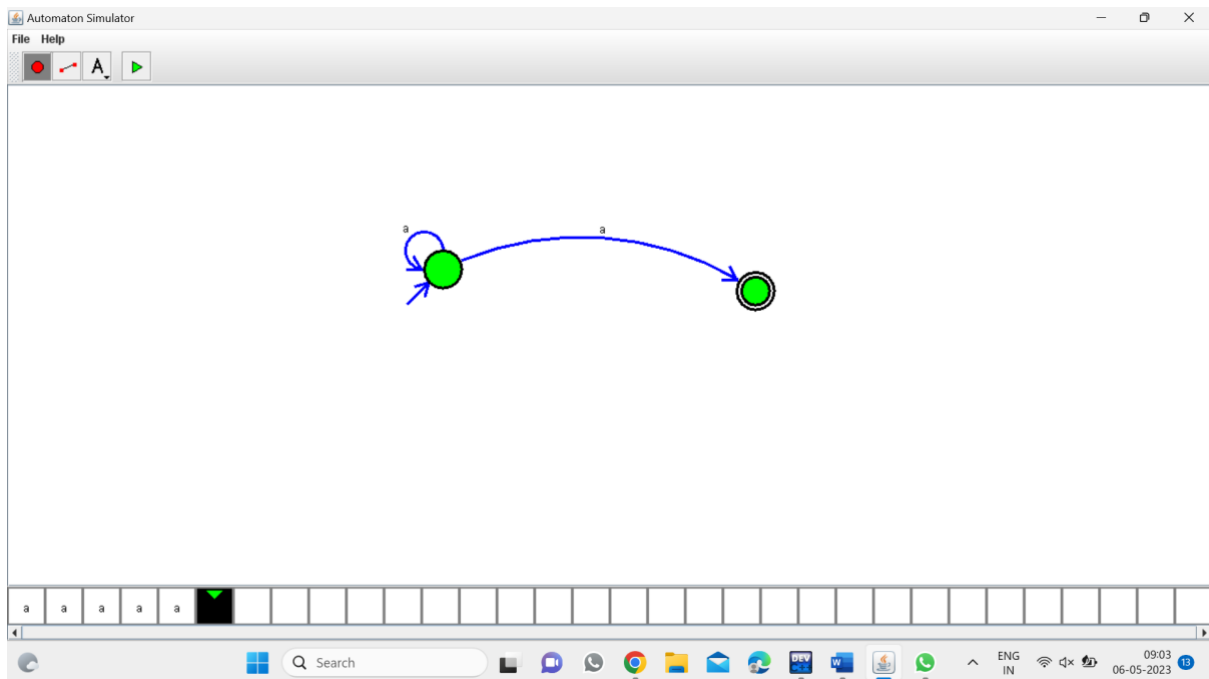
### PROGRAM12: 29

### DFA to accept bcaaaaaa:



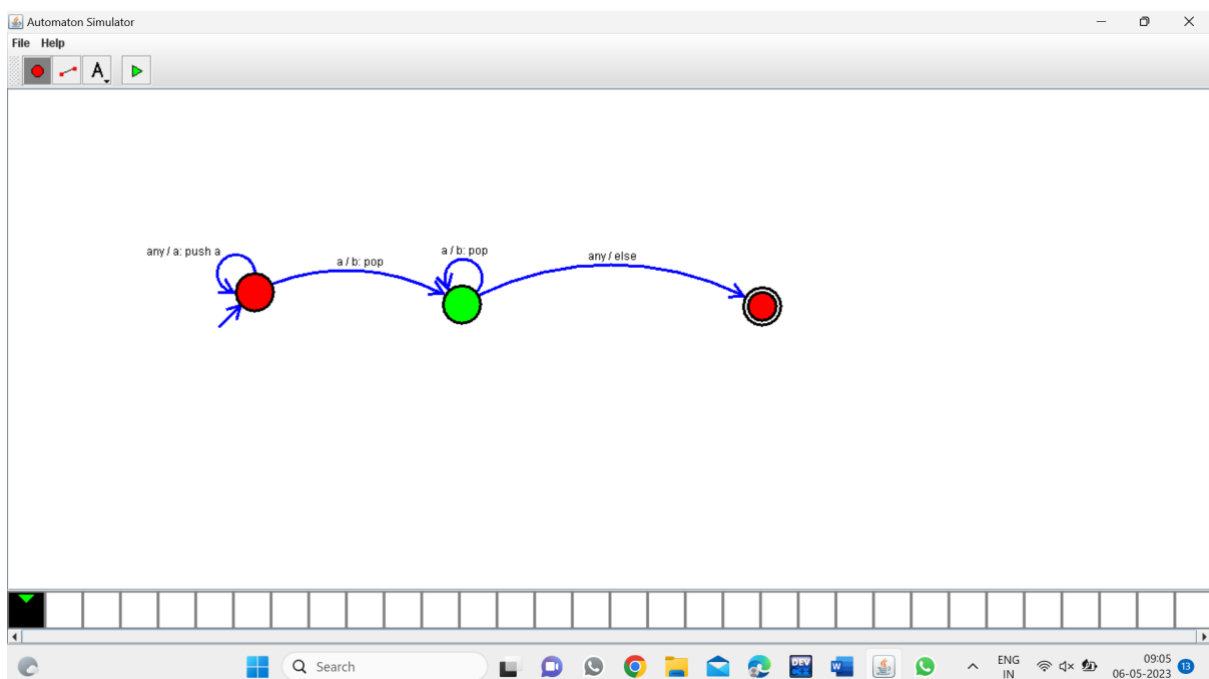
## PROGRAM13:

**NFA to accept aaaa: 30**



## PROGRAM14: 31

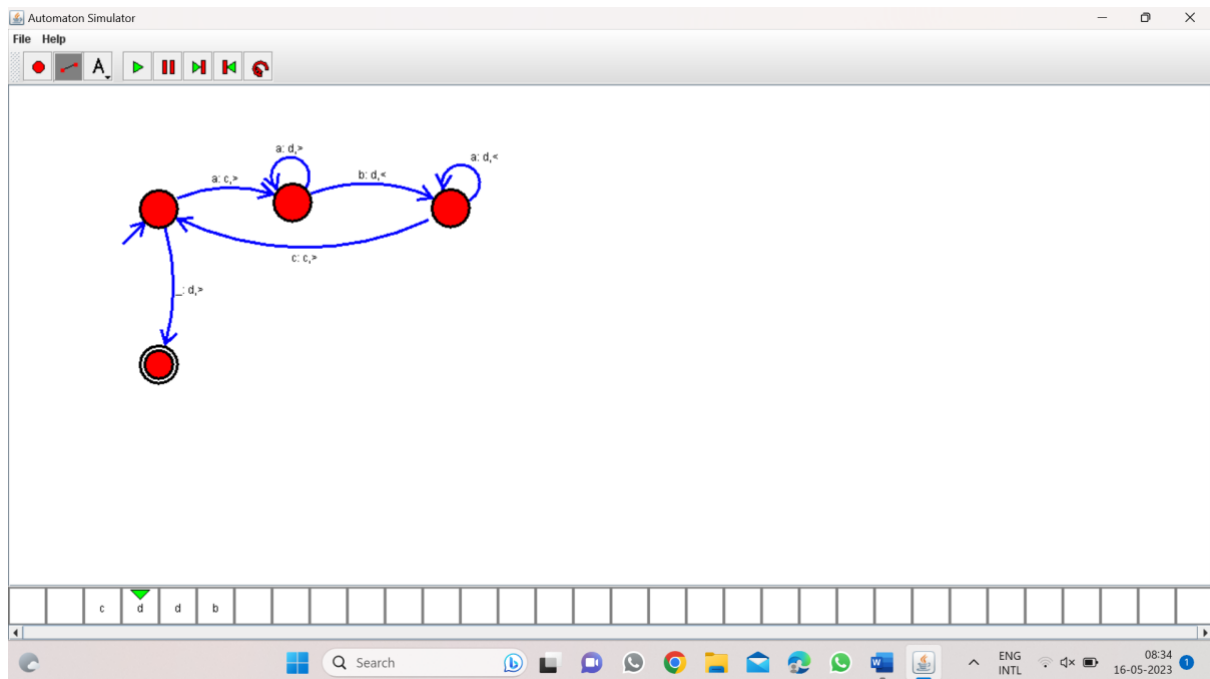
**PDA for  $a^n b^n$ :**



## PROGRAM 15:

TM For  $a^n b^n$ :

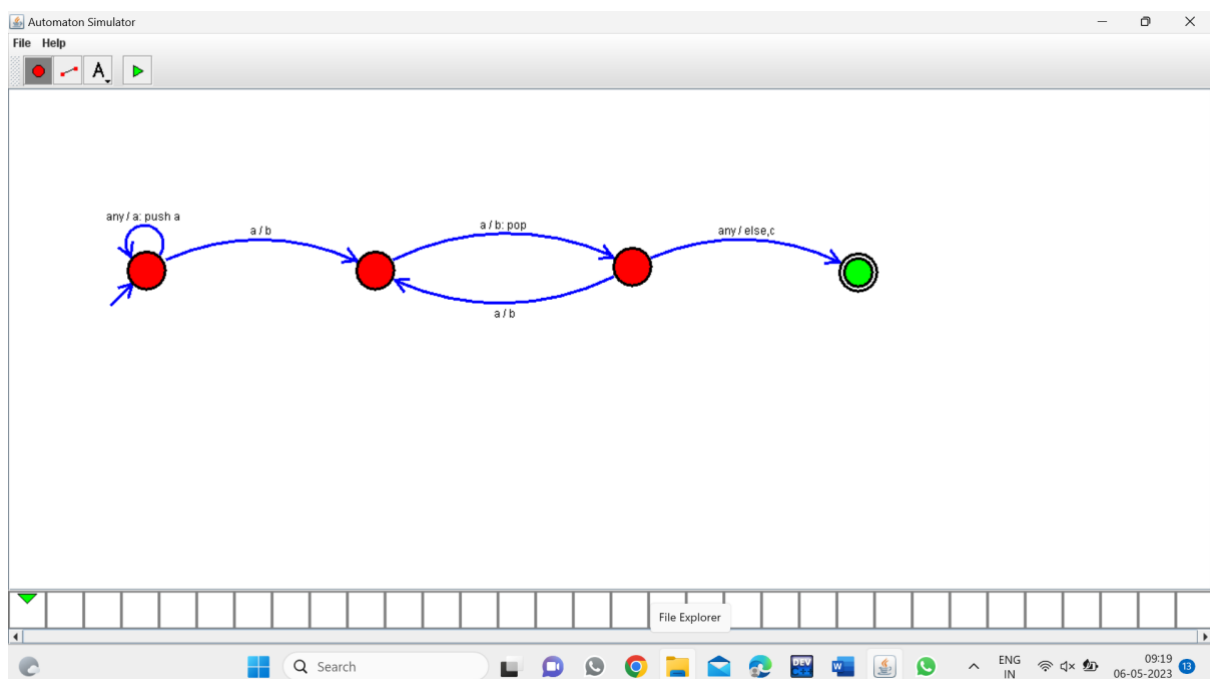
15



## PROGRAM16:

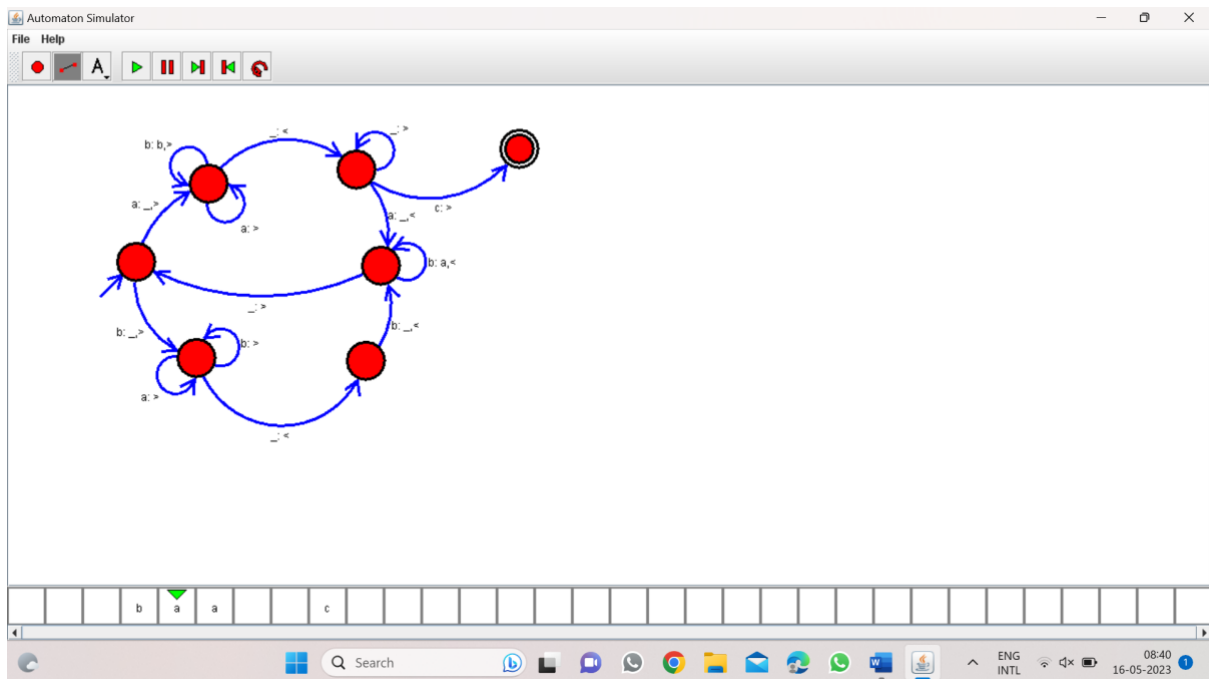
16

TM for  $a^n b^{2n}$ :



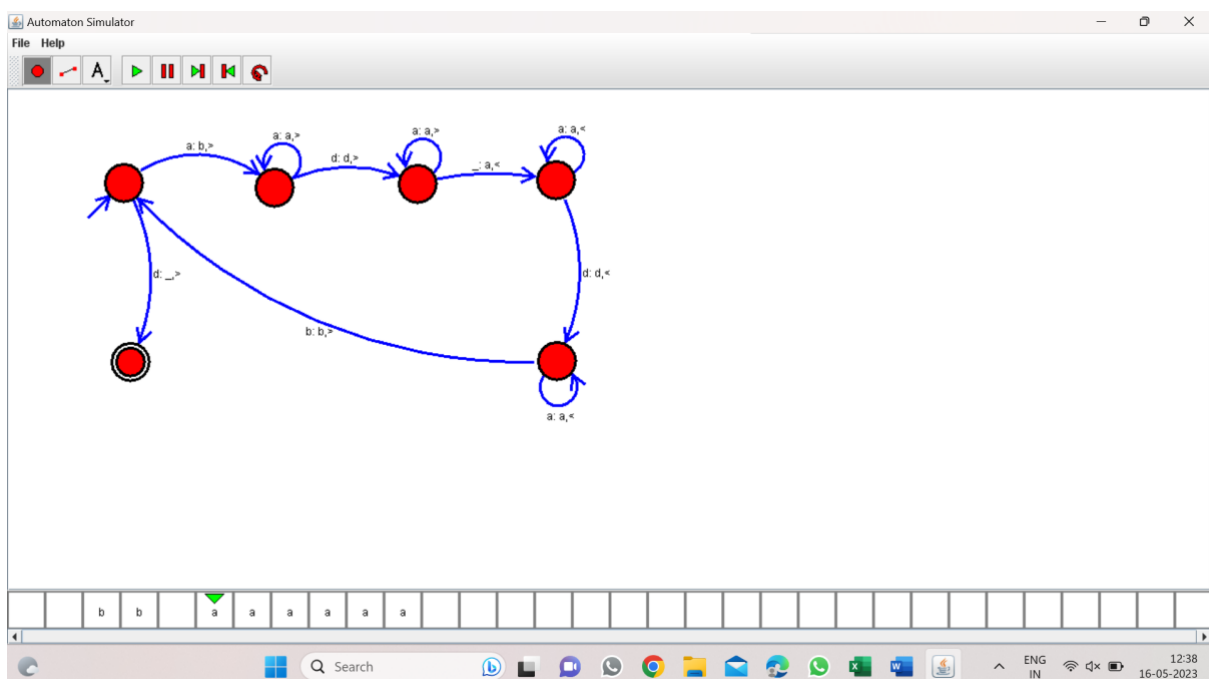
## Program17: 17

### TM for palindrome



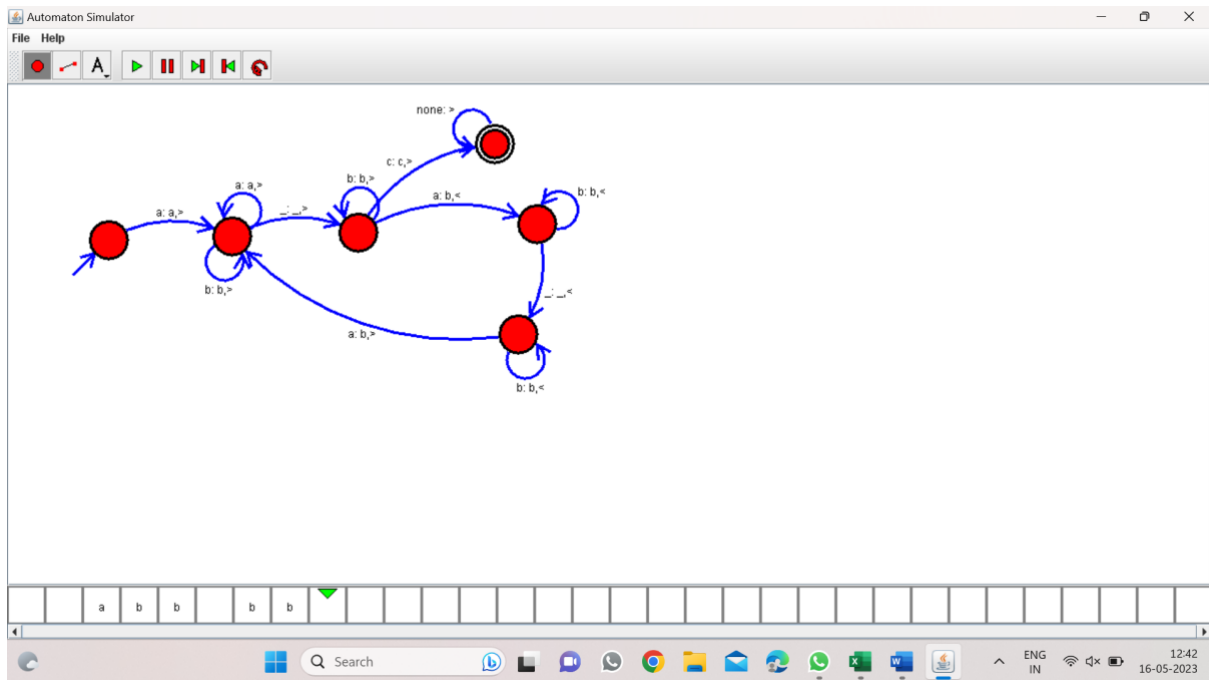
## Program 18: 19

### TM for addition:



20

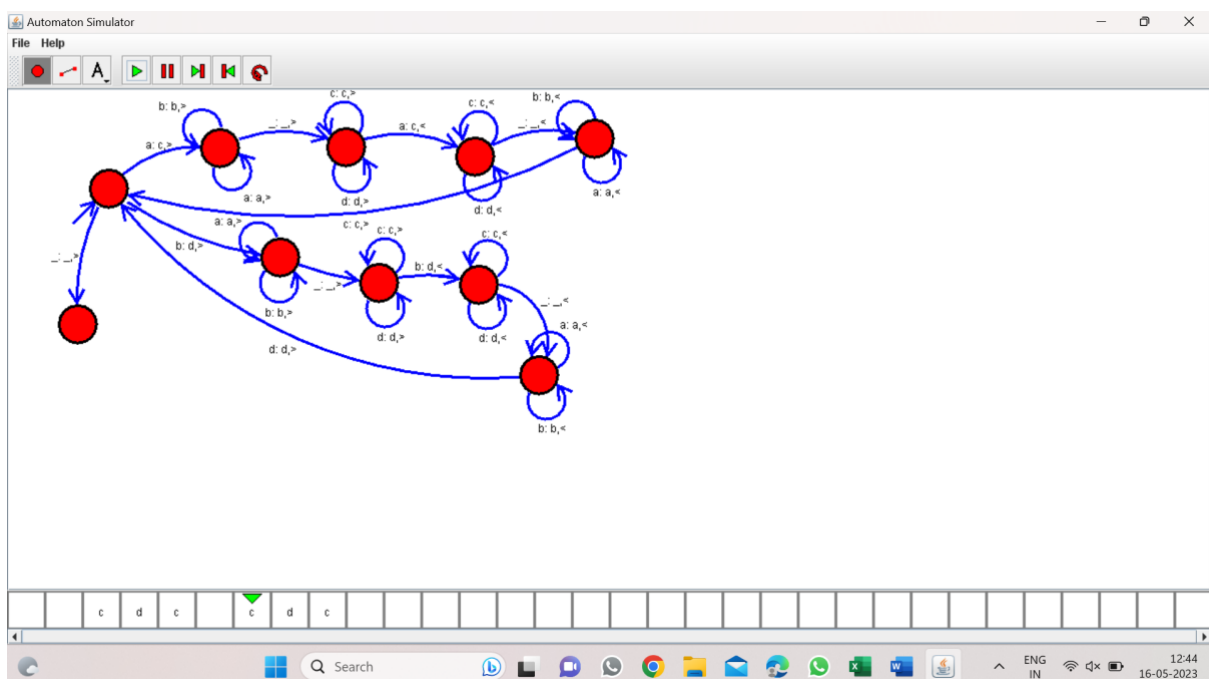
### TM for subtraction:



## Program 20:

32

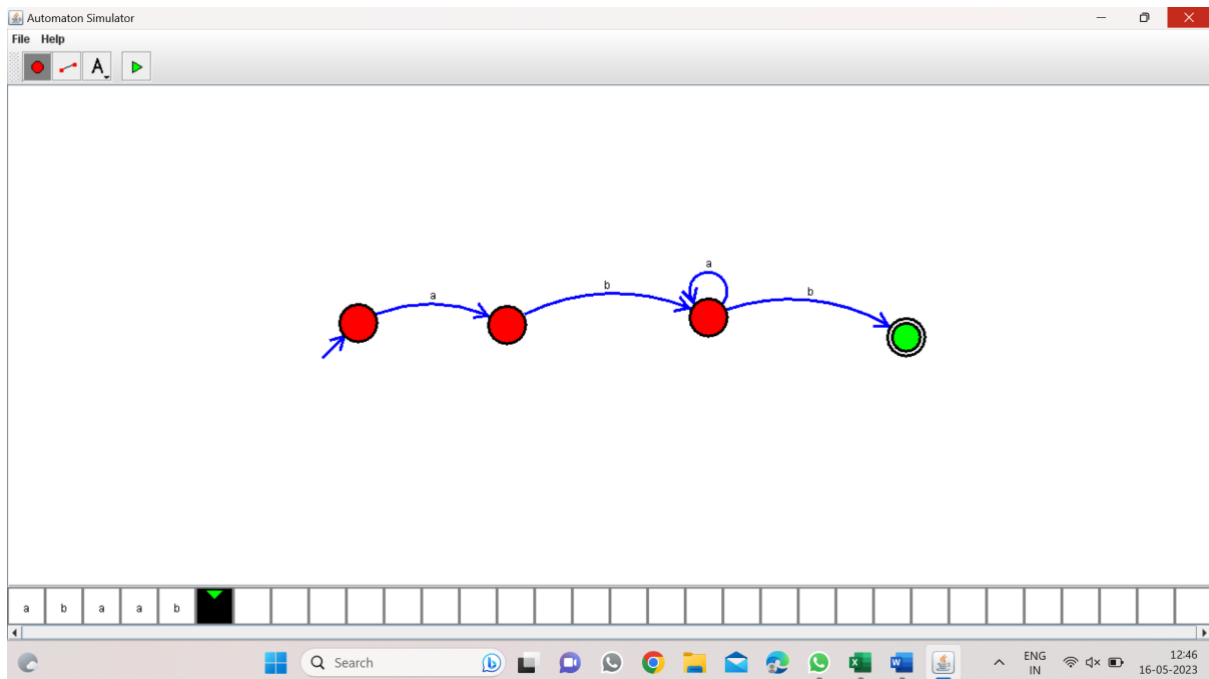
### TM for string coparision:



**Program 21: 36**

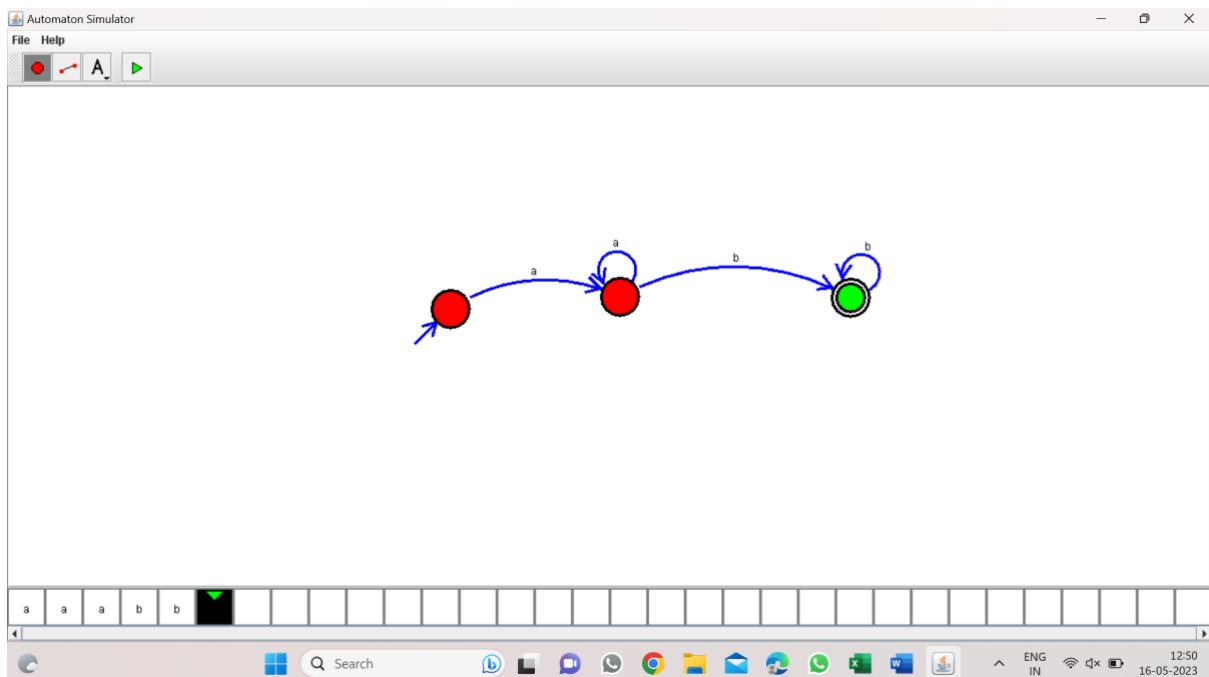
**NFA to accept start with a and end with b:**

**W=abaab**



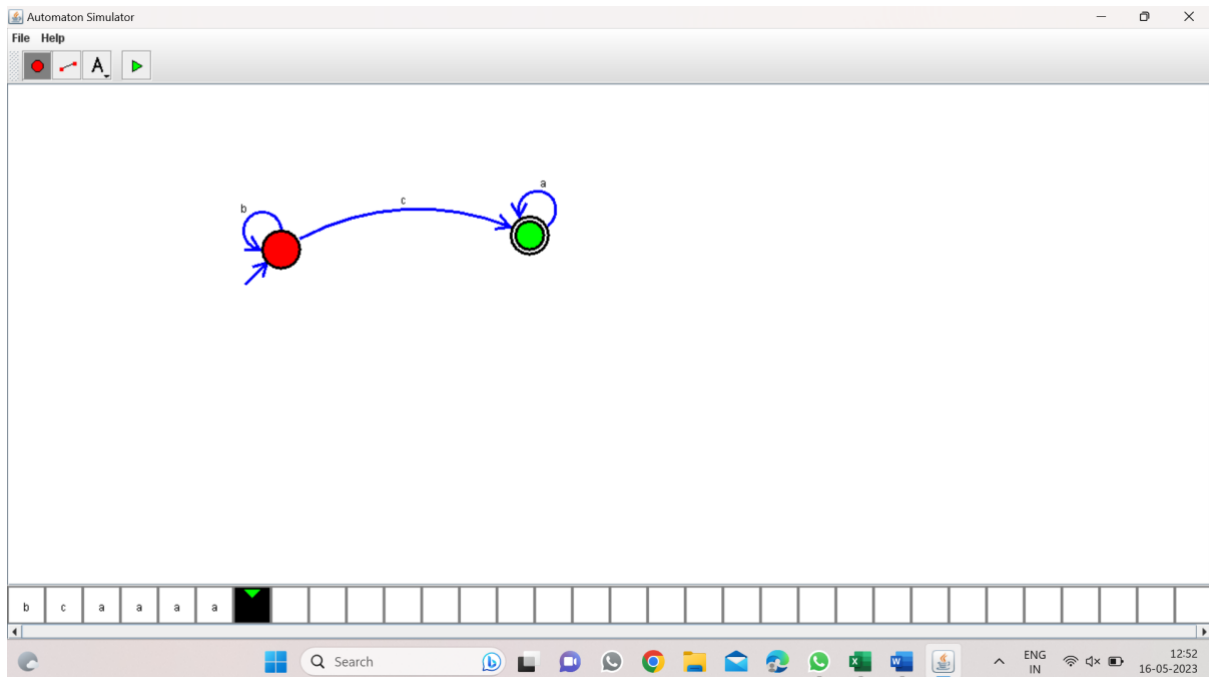
**Program 22: 37**

**NFA to accept string that start and end with different symbols:**



## Program 23: 38

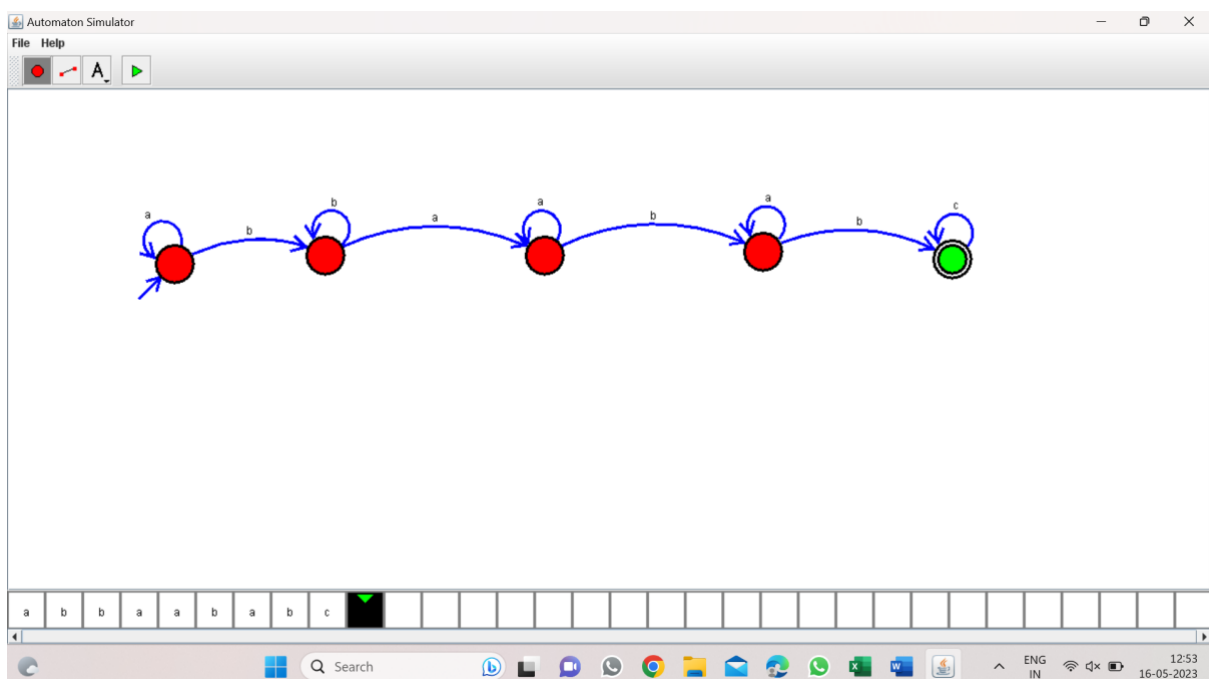
**NFA to accept the string bbc,c,bcaaa:**



## Program 24: 39

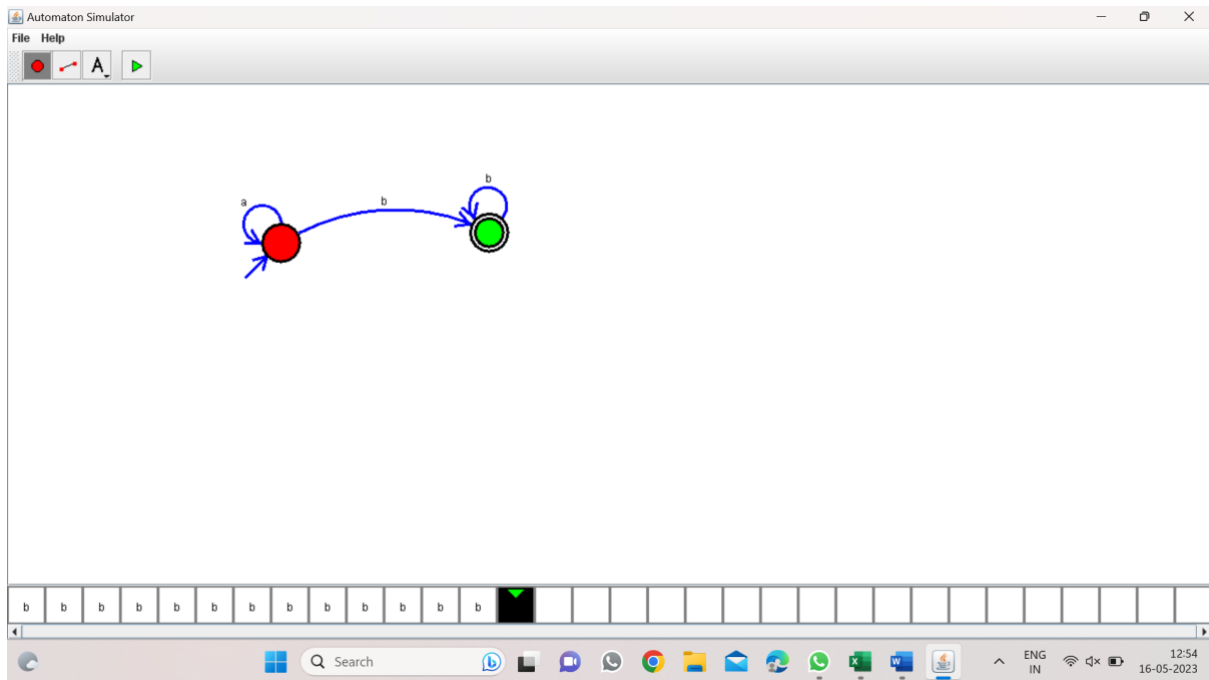
**DFA to accept the string that end with abc:**

**W=abbaababc**



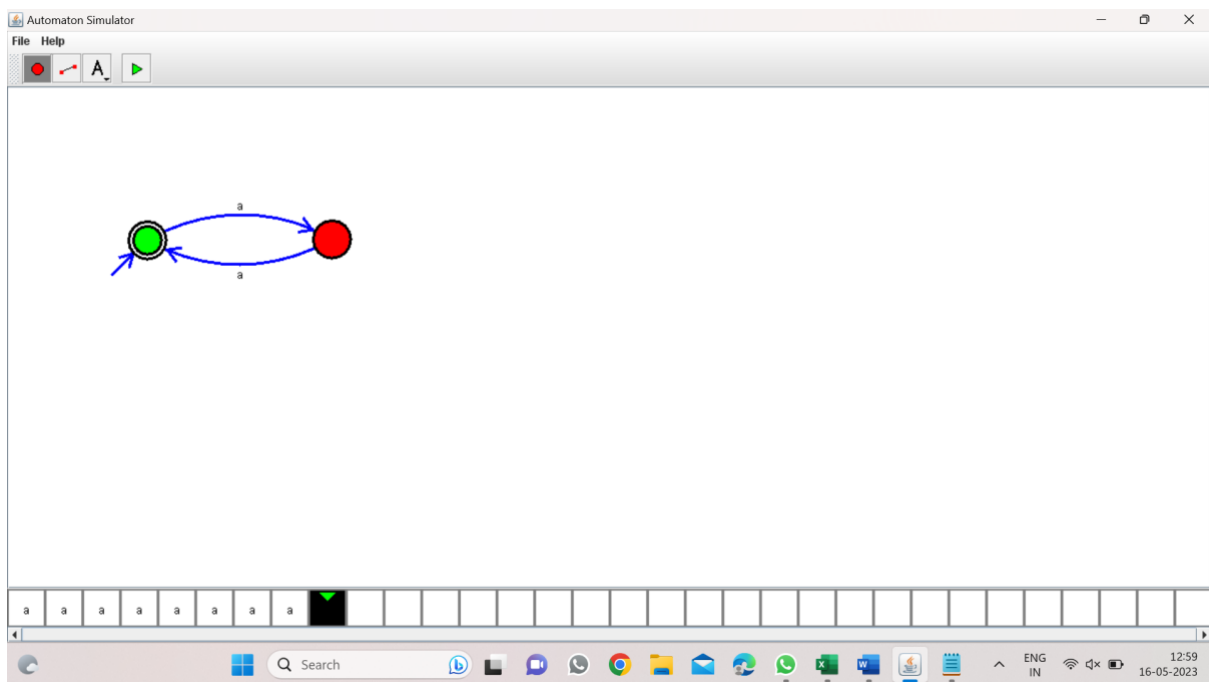
## Program 25: 40

**NFA to accept any number of b's:**



## Program 26: 21

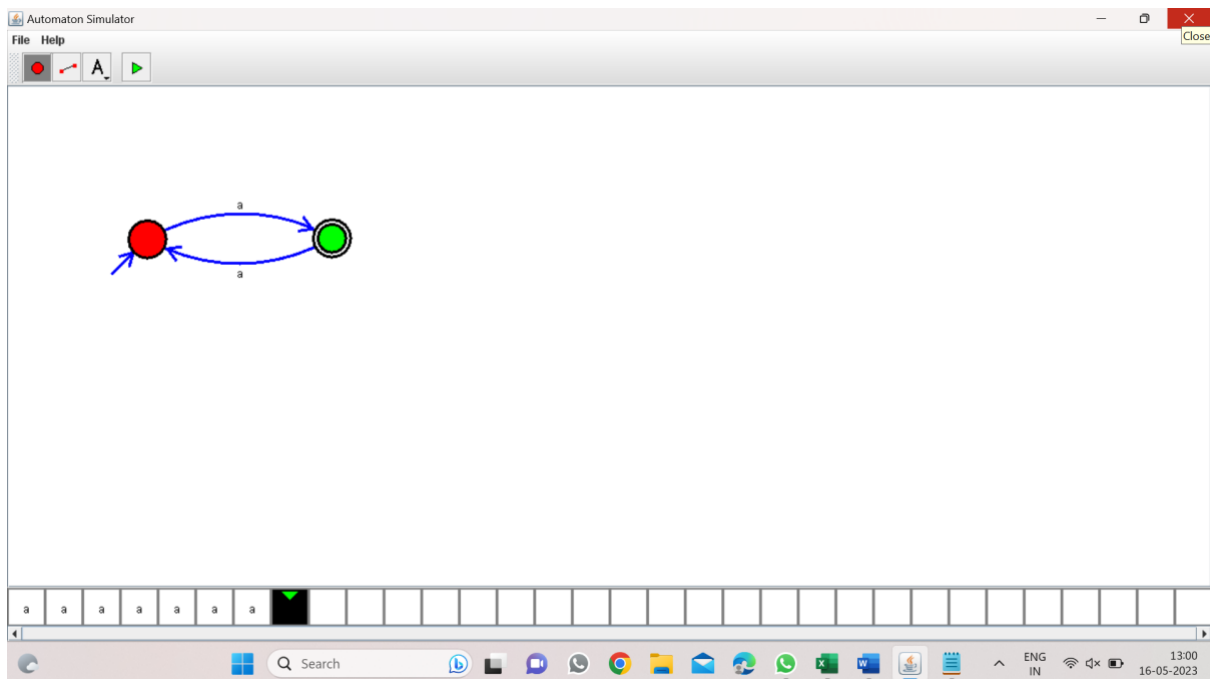
**DFA for even no.of a's:**





## Program27: 22

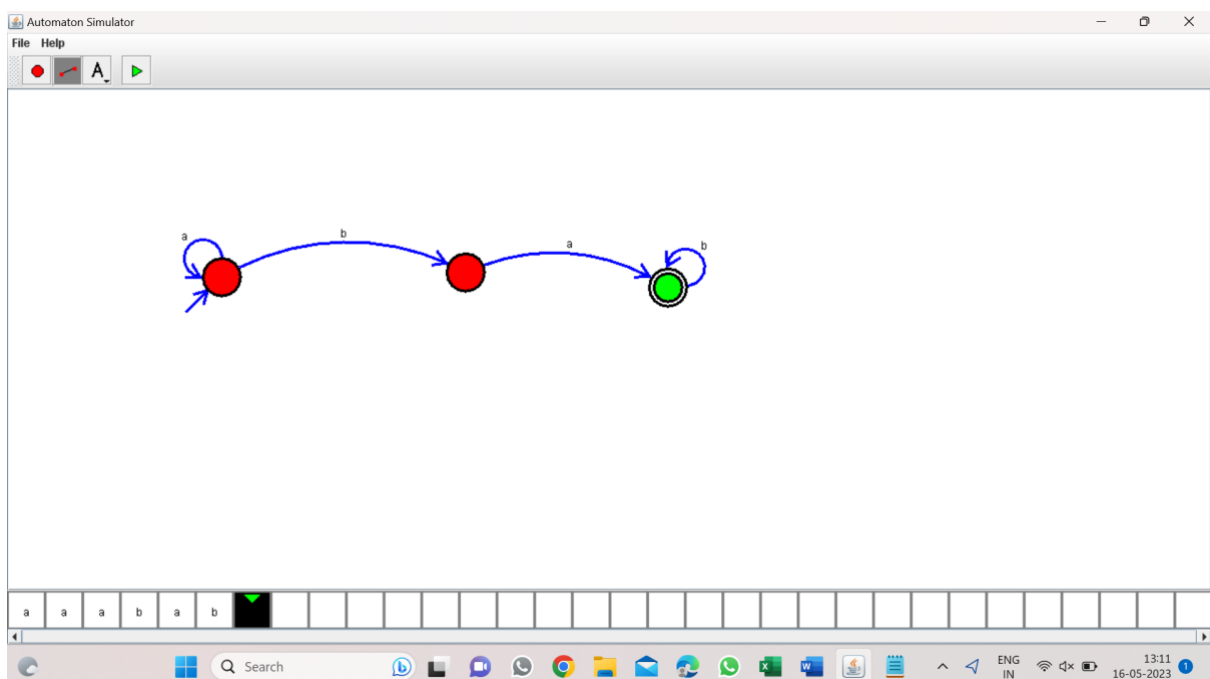
**DFA for odd no.of a's:**



## Program 28: 23

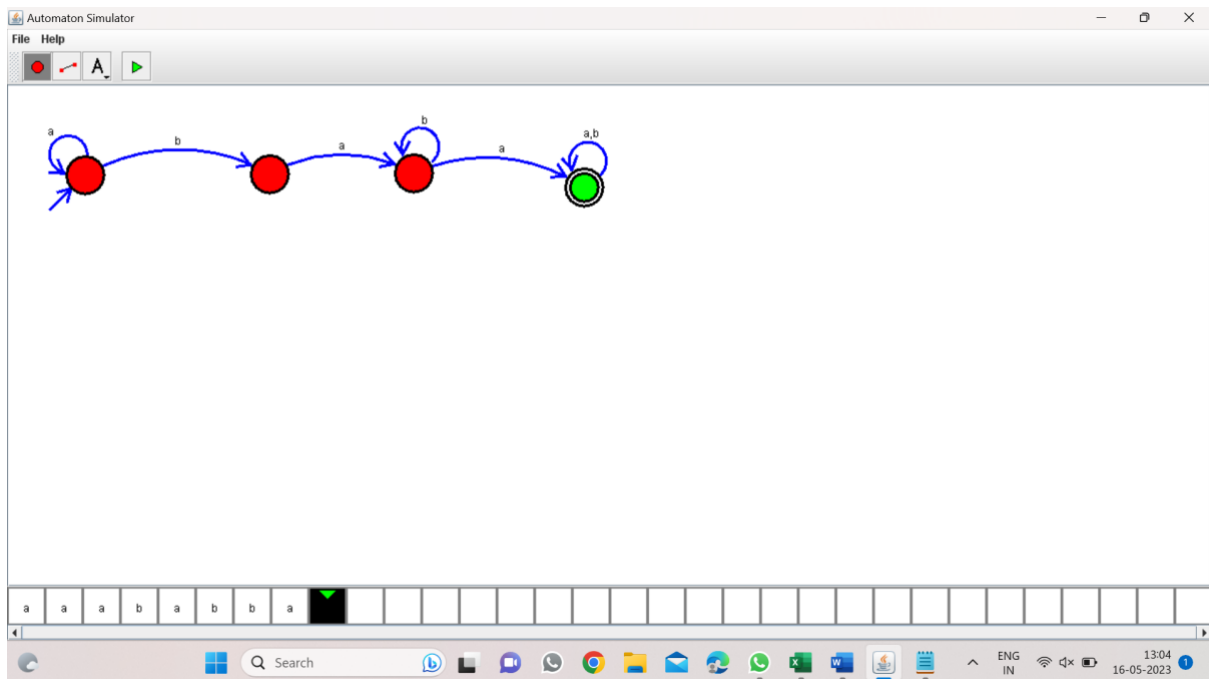
**DFA to accept string end with ab:**

**W=aaabab:**



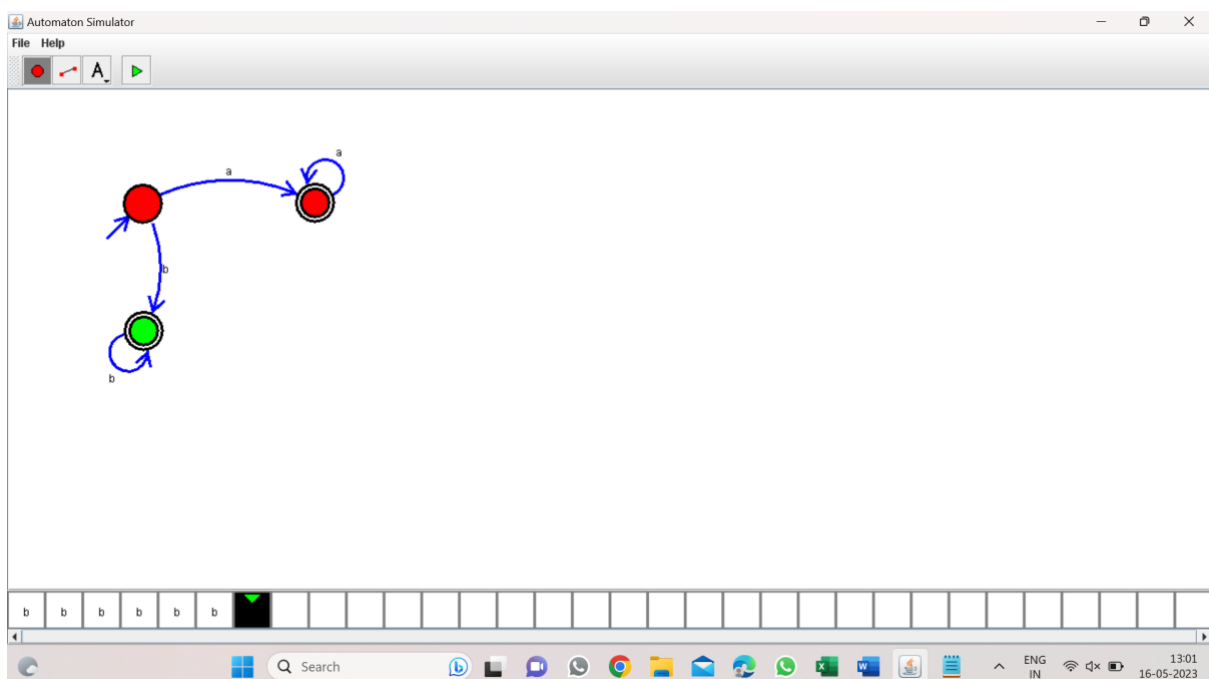
## Program 29: 24

**DFA consisting substring ab:**



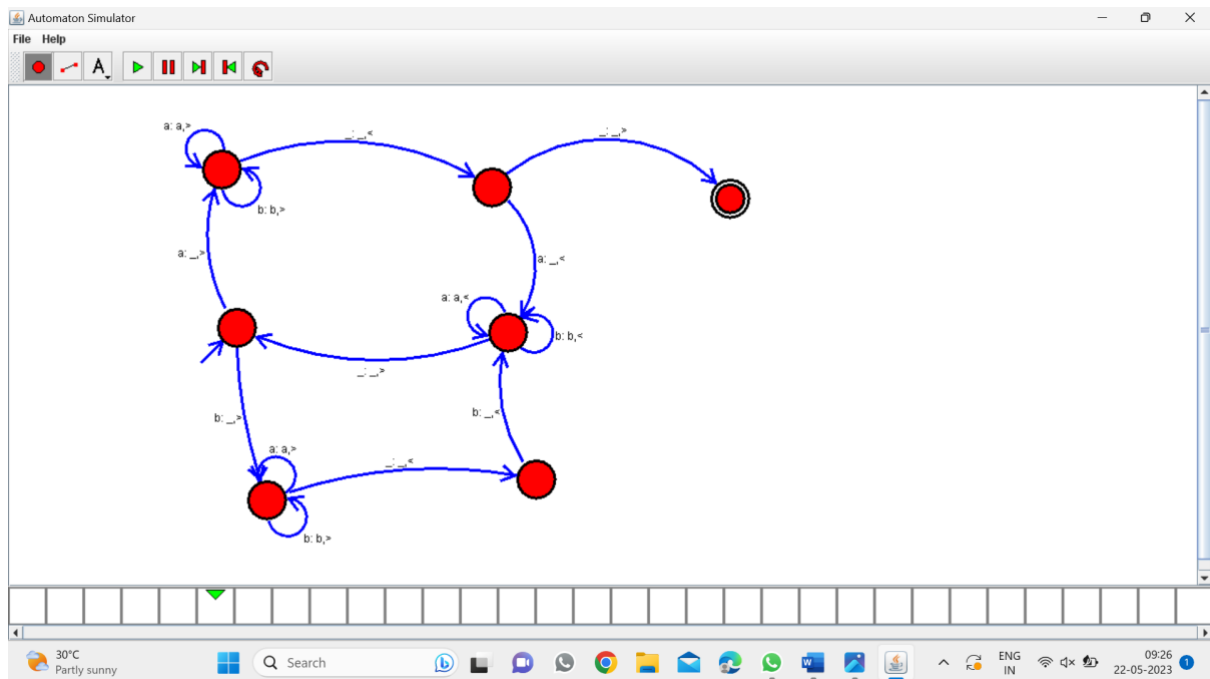
## Program 30: 25

**DFA start with a or b:**



## PROGRAM 31: 26

### TM TO ACCEPT STRING PALINDROME (bbabb):



## PROGRAM 32:

### DFA TO ACCEPT EVEN NUMBER OF C's: