**1.DETERMINISTIC FINITE AUTOMATA**

#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 the 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");

}

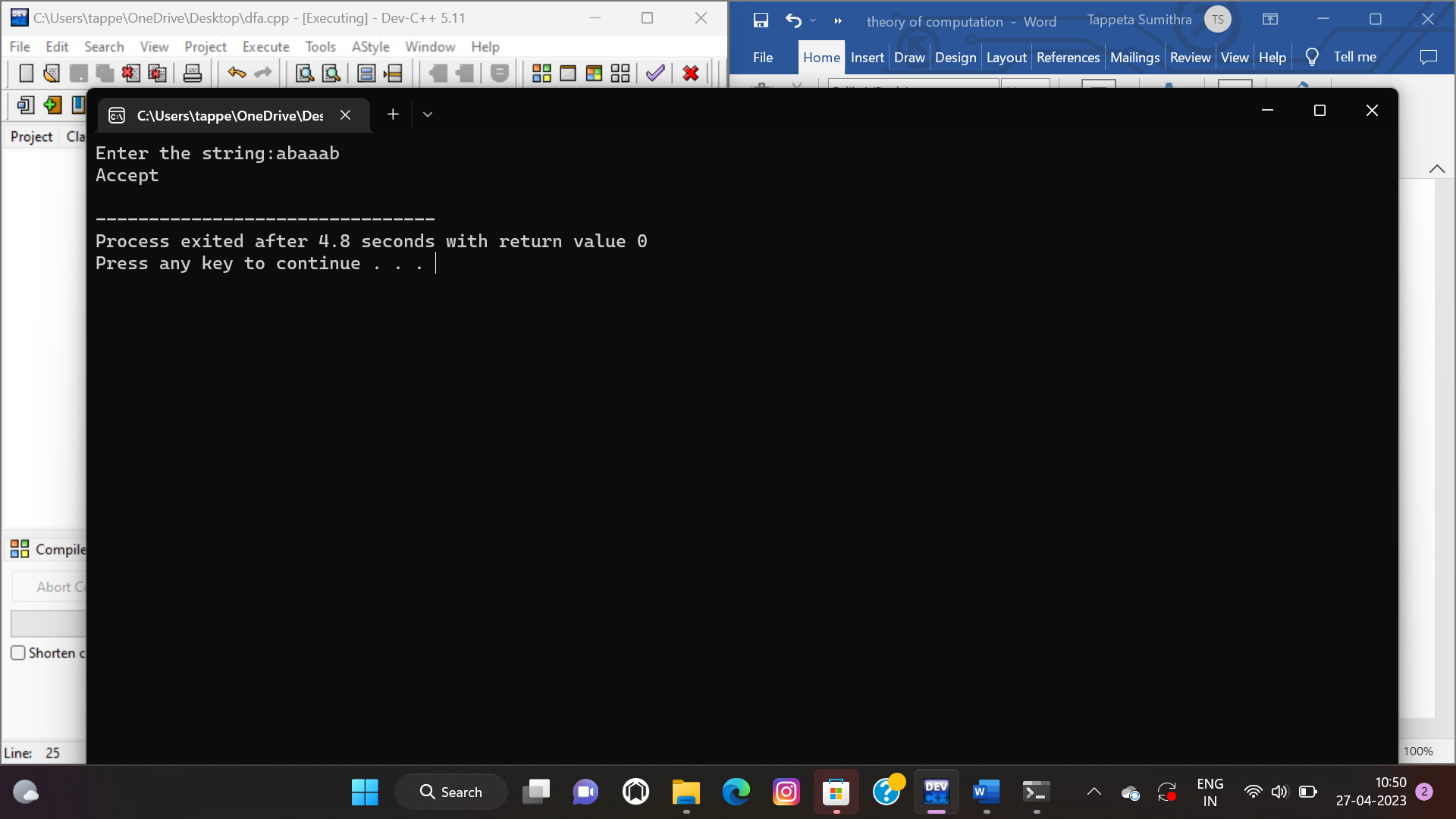
else if(present\_state==final\_state)

printf("Accept\n");

else

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

}



**2.NON-DETEMINISTIC FINITE AUTOMATA**

#include<stdio.h>

#include<string.h>

#include<conio.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 alphabeta : ");

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 shown 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')

inp1=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");

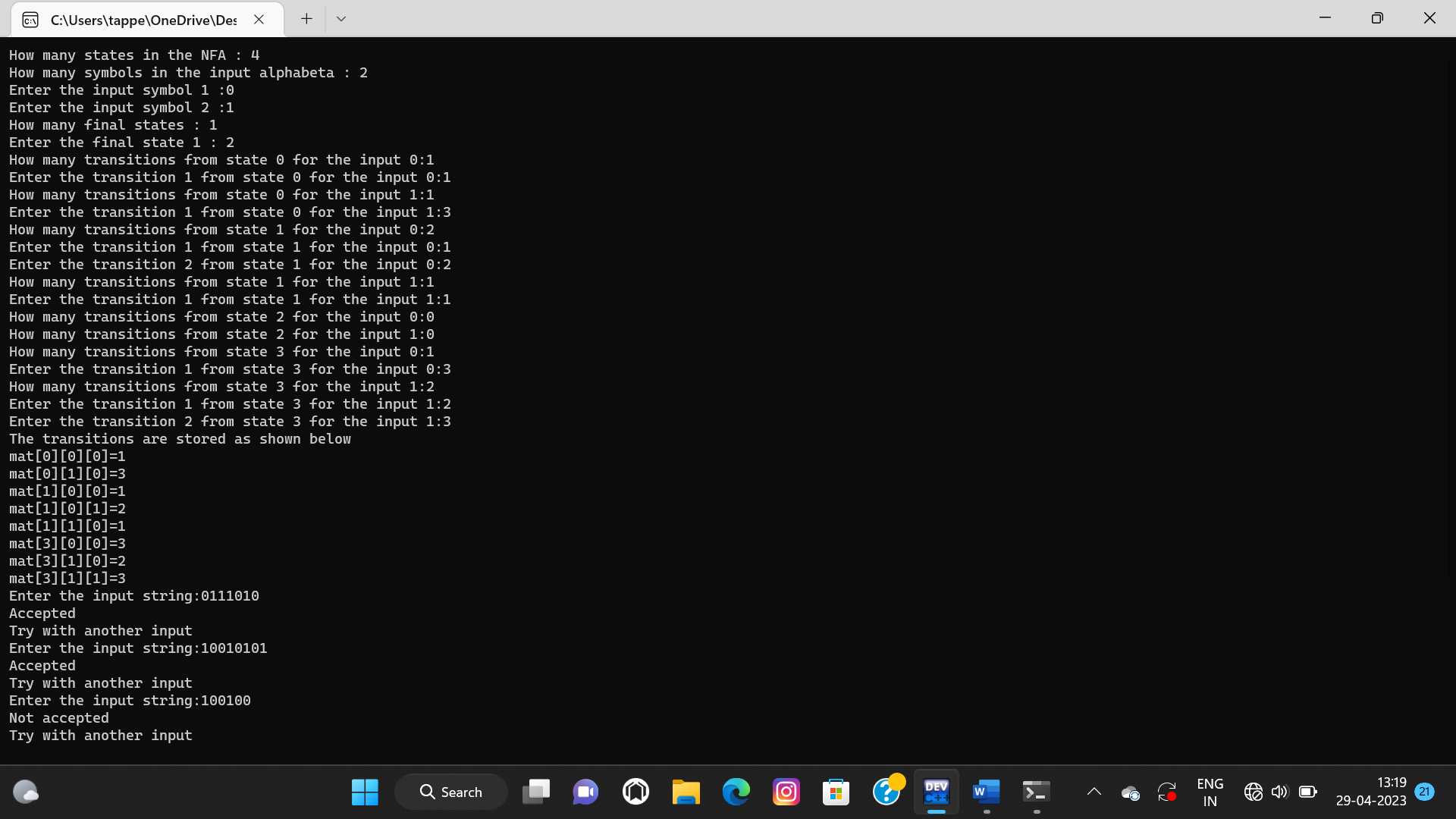
else

printf("Not accepted\n");

printf("Try with another input\n");

}

}



**3.EPS-CLOSURE FOR NFA**

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++)

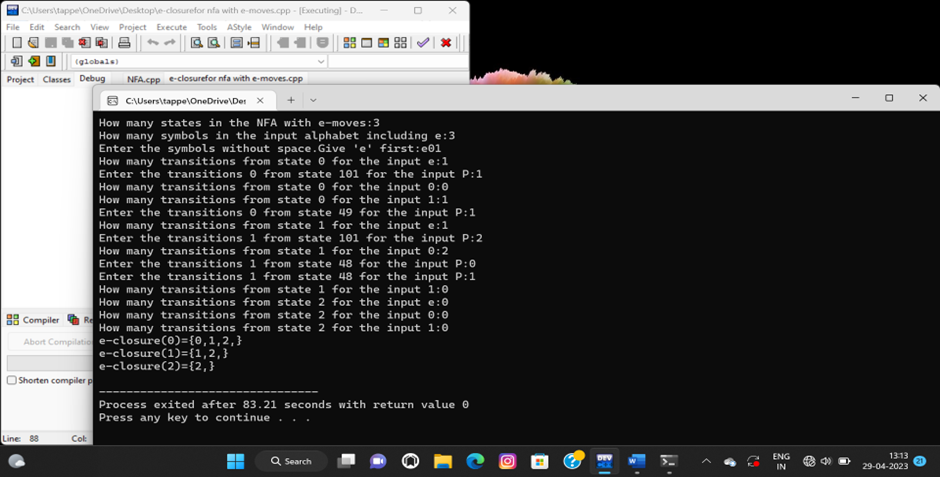
{

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

ptr++;

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

}

}

**4.CHECKING WHETHER A STRING BELONGS TO A GRAMMAR OR NOT**

#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')

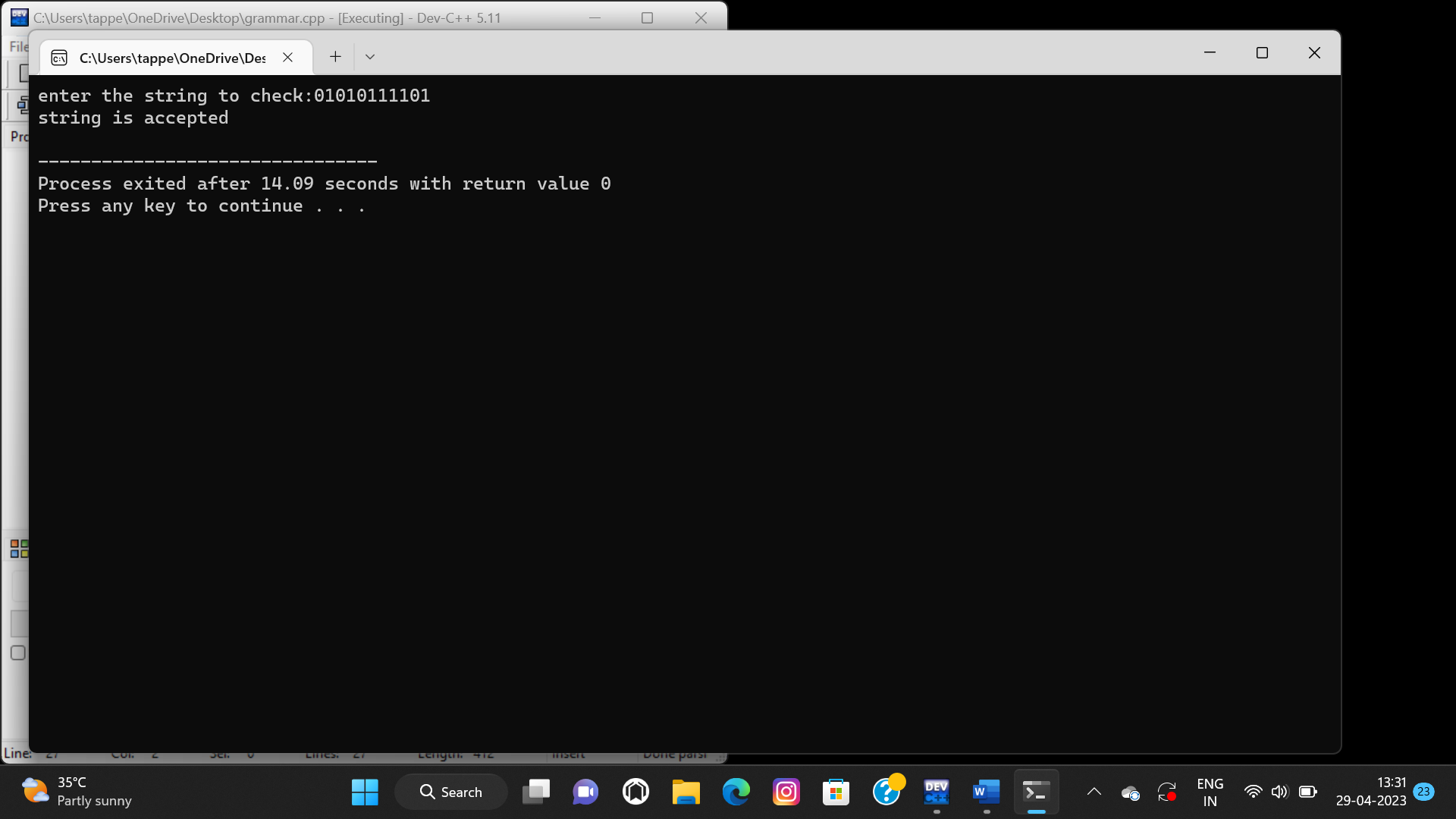
printf("string is accepted\n");

else

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

}

}



**5.CHECKING WHETHER A STRING BELONGS TO GRAMMAR OR NOT**

#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!=(1/2))

{

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

{

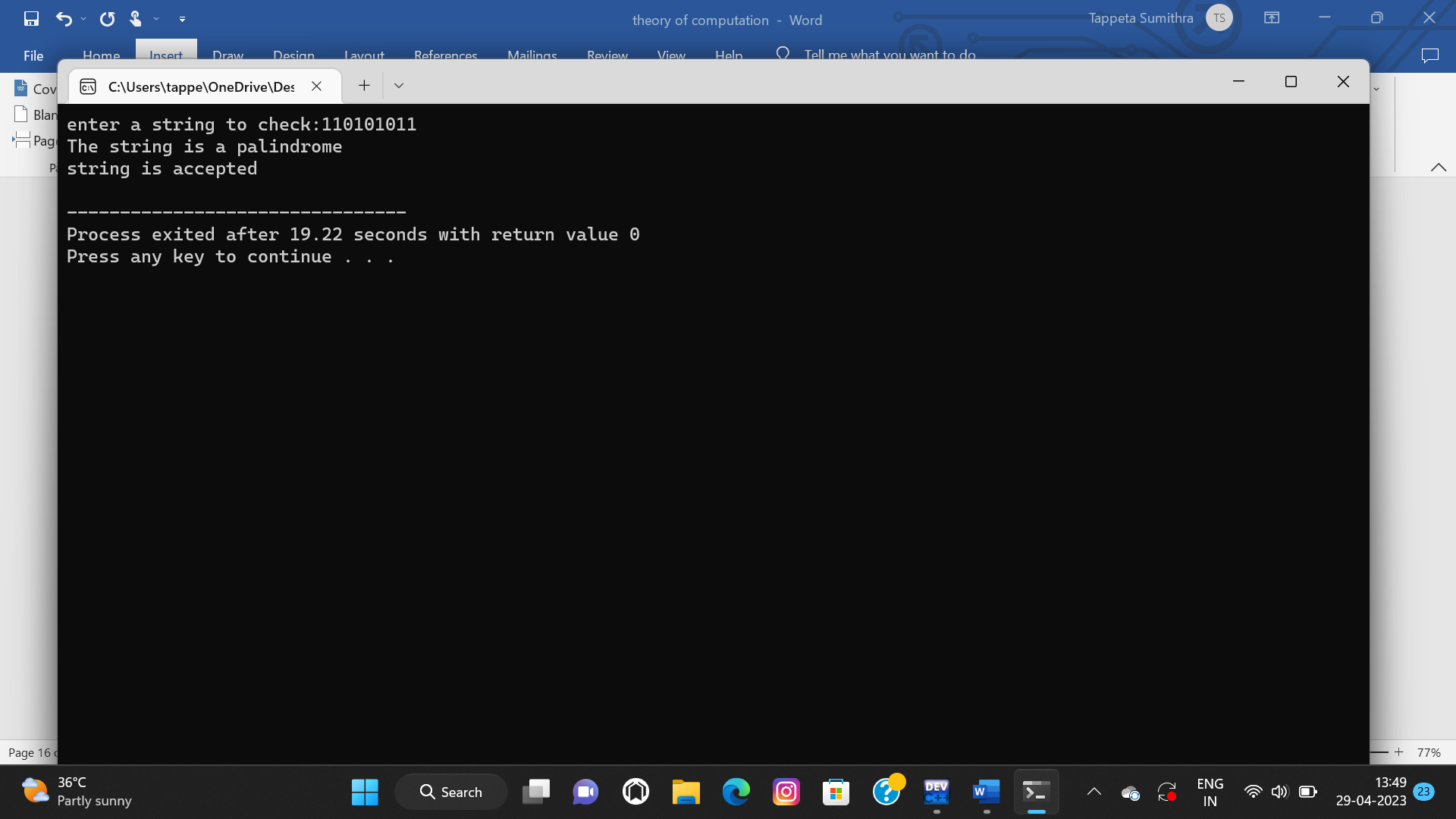
printf("The string is not a palindrome\n");

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

}

}

}



**6.CHECKING WHETHER A STRING IS GRAMMAR OR NOT**

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,a,b;

int l,count1,count2;

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)

{

i=0;count1=0;

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

{

count1++;

i++;

}

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

{

i++;

}

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

{

i++;

}

flag1=1;

count2=0;

while(i<1)

{

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");

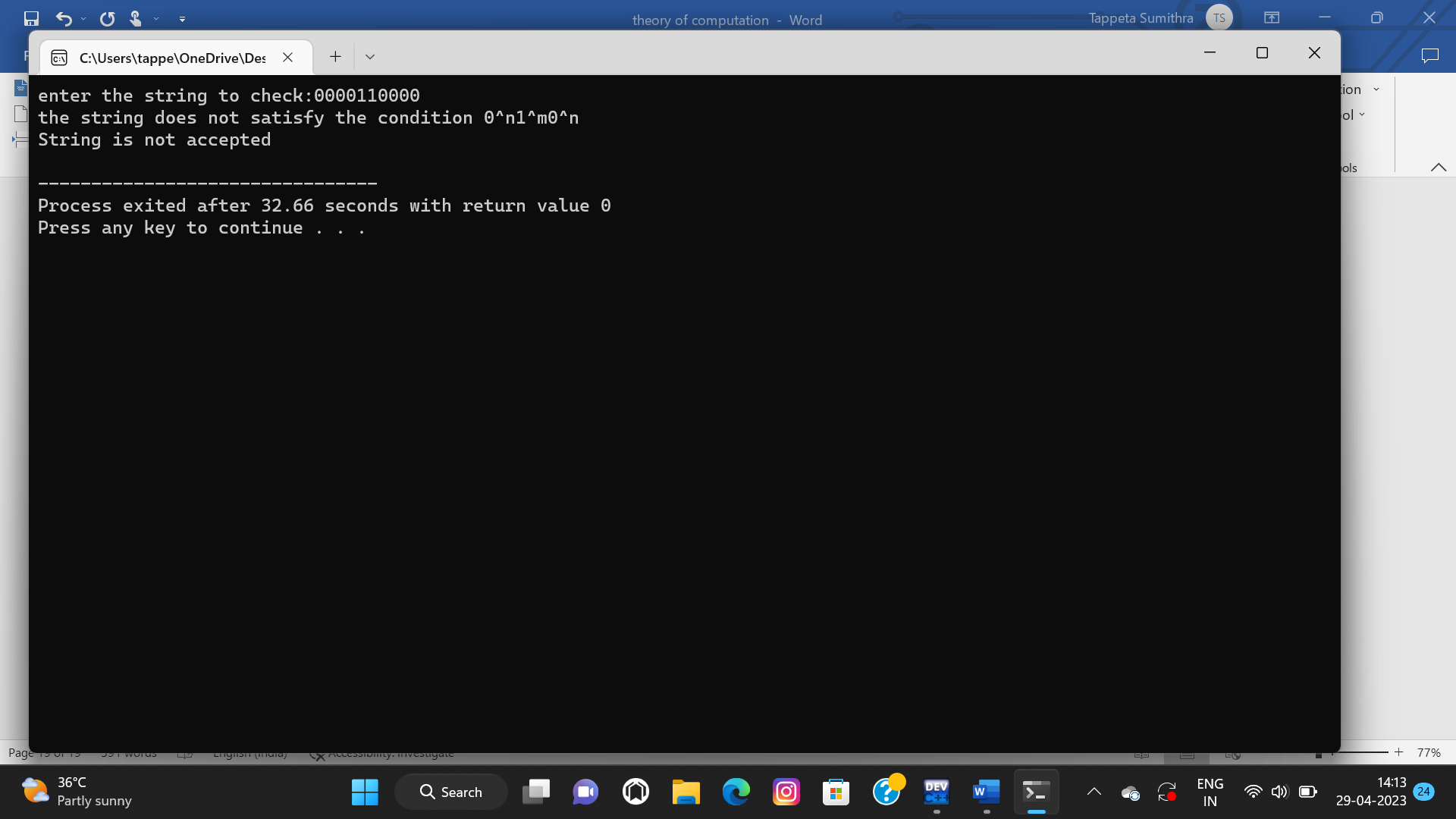
printf("String is not accepted\n");

}

}

}

}



**7.CHECKING WHETHER A STRING BELONGS TO GRAMMAR OR NO**

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,flag2;

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(l%2!=0)

{

printf("the string does not satisfy the conditon 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<1;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");

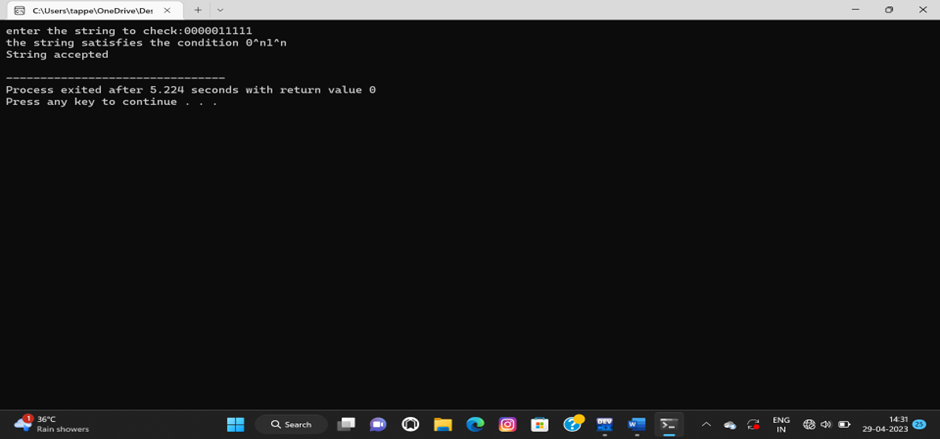
printf("String not accepted\n");

}

}

}

}



**8.CHECKING WHETHER A STRING BELONGS TO GRAMMAR OR NOT**

#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 accepted\n");

break;

}

}

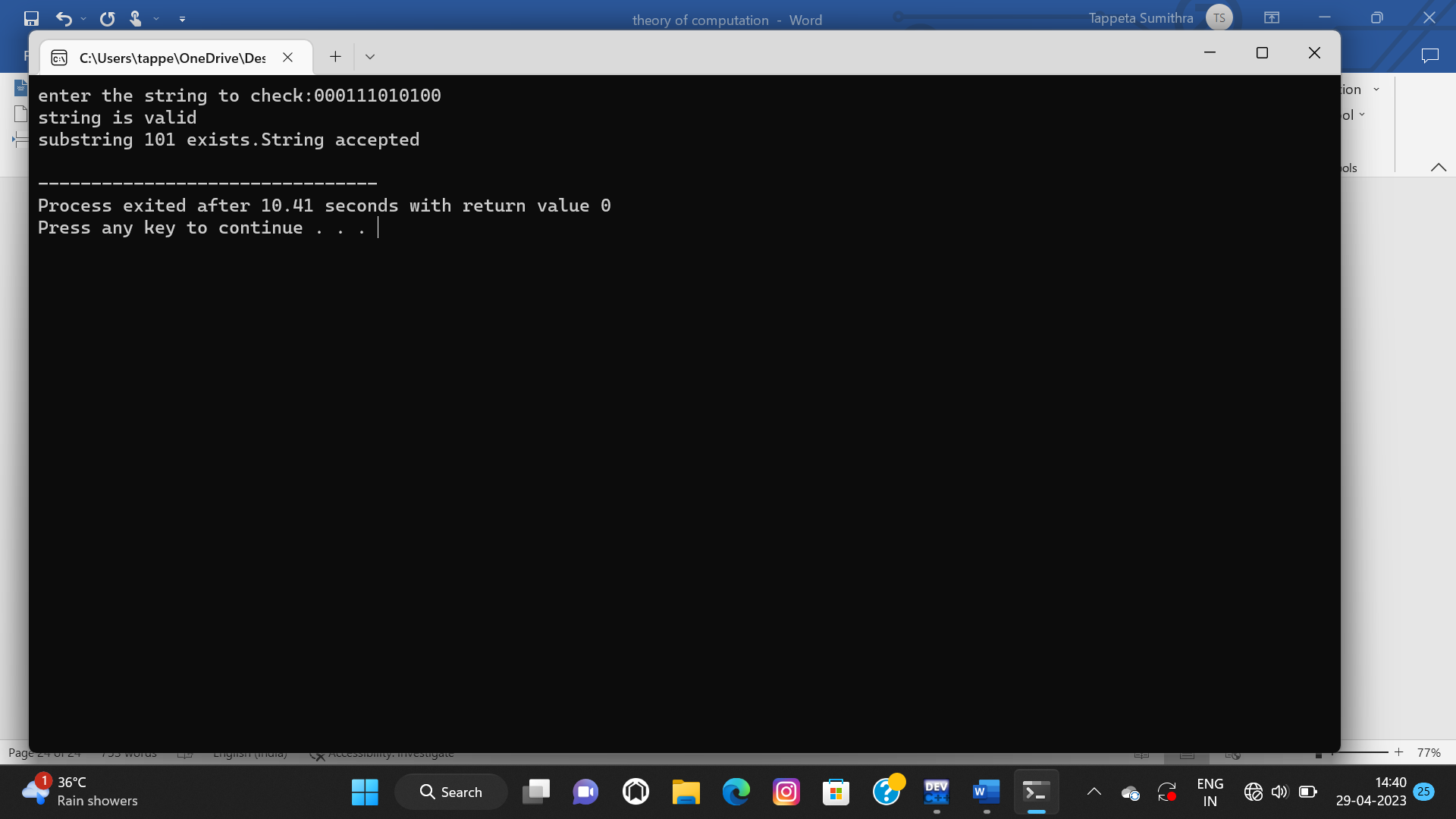
}

if(flag1==0)

printf("substring 101 does not exist.String not accepted\n");

}

}



**9.Pushdown automata**

**#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==(l))**

**{ break;**

**}**

**}**

**if(top>=1)**

**{**

**printf("String not accepted");**

**} else**

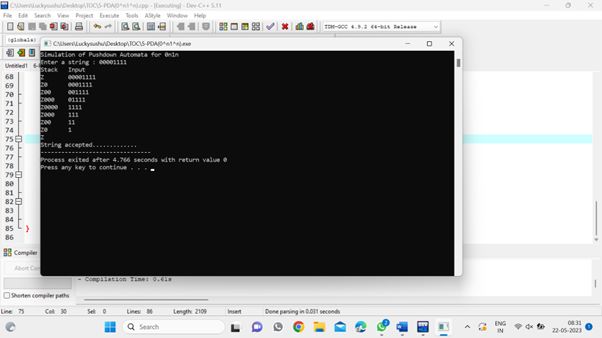
**{**

**printf("String accepted");**

**} b: printf(".............");**

**}**

**Output:**



**10.pushdown automata**

**#include<stdio.h>**

**#include<string.h>**

**int main()**

**{**

**int i,j,le,flag,flag1,flag2;**

**char str[20];**

**printf("Program to show how a turing machine will process 0n1n2n\n");**

**printf("Enter a string : ");**

**scanf("%s",str);**

**le=strlen(str);**

**j=0; while(1)**

**{**

**flag=0;flag1=0;**

**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)**

**{**

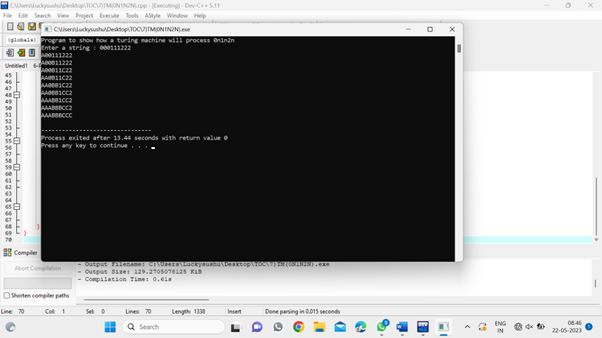
**break;**

**}**

**}**

**}**

**Output:**



**11.Turning machine**

**#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==l)**

**{ break;**

**}**

**}**

**if(top>=1)**

**{**

**printf("String not accepted");**

**} else**

**{**

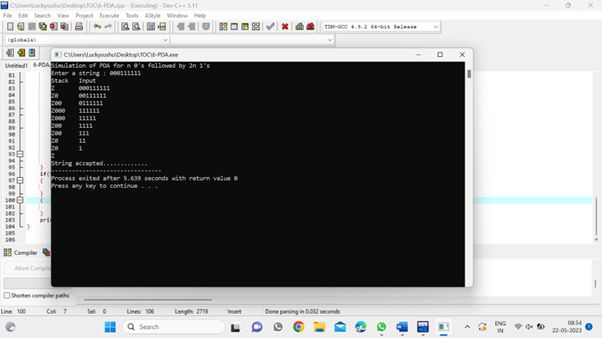
**printf("String accepted");**

**} b:**

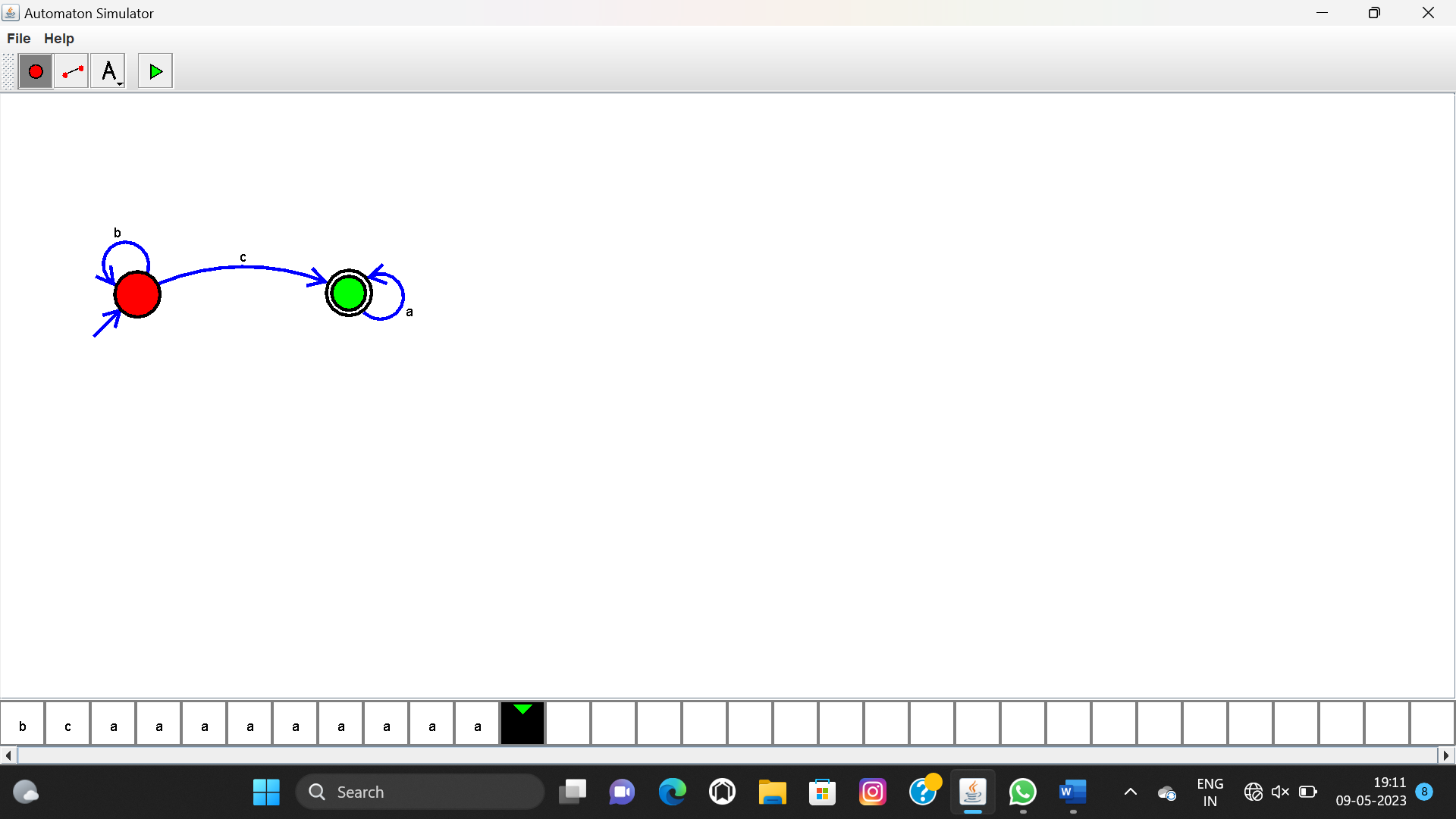
**printf(".............");**

**}**

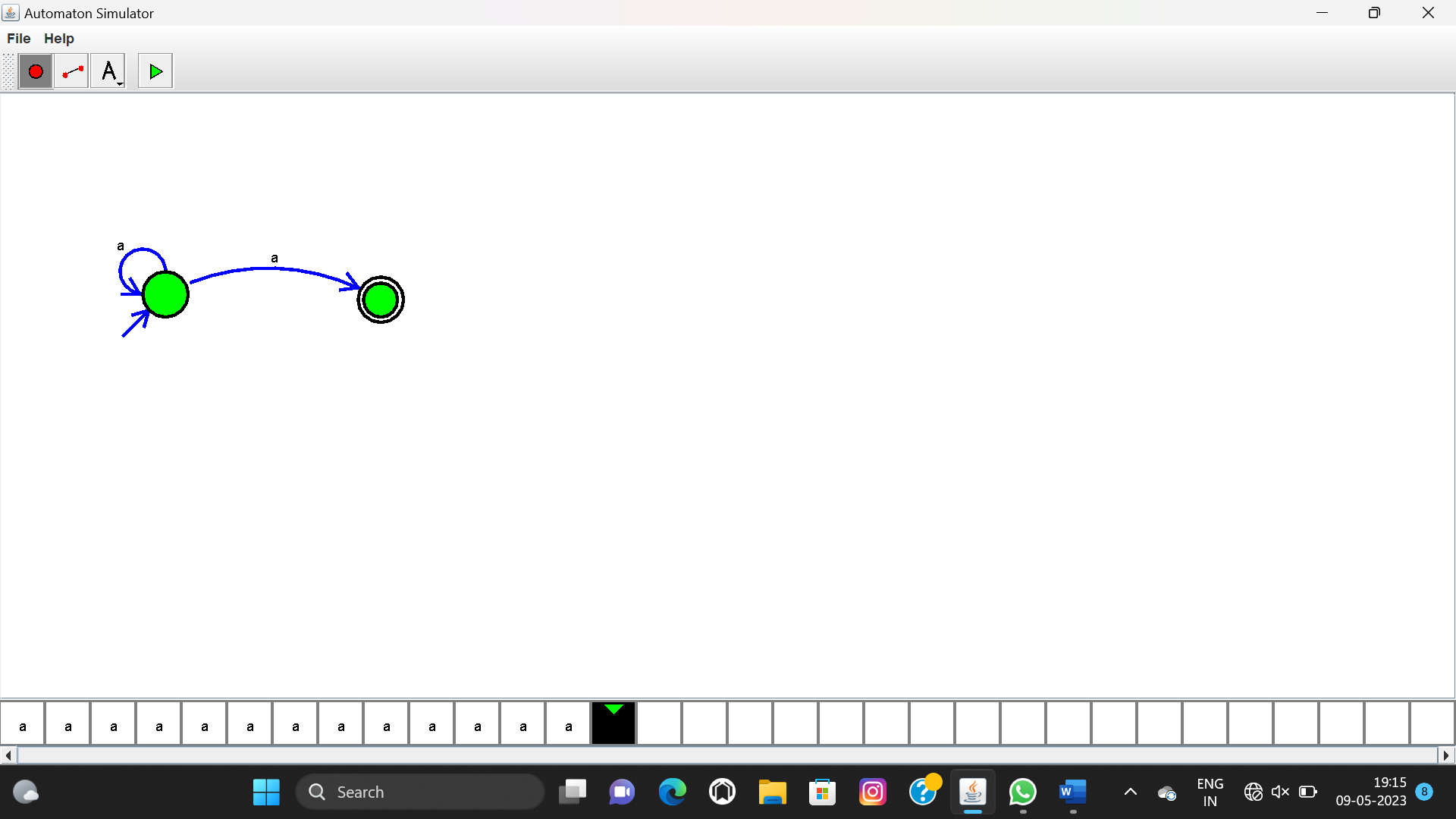
**Output:**



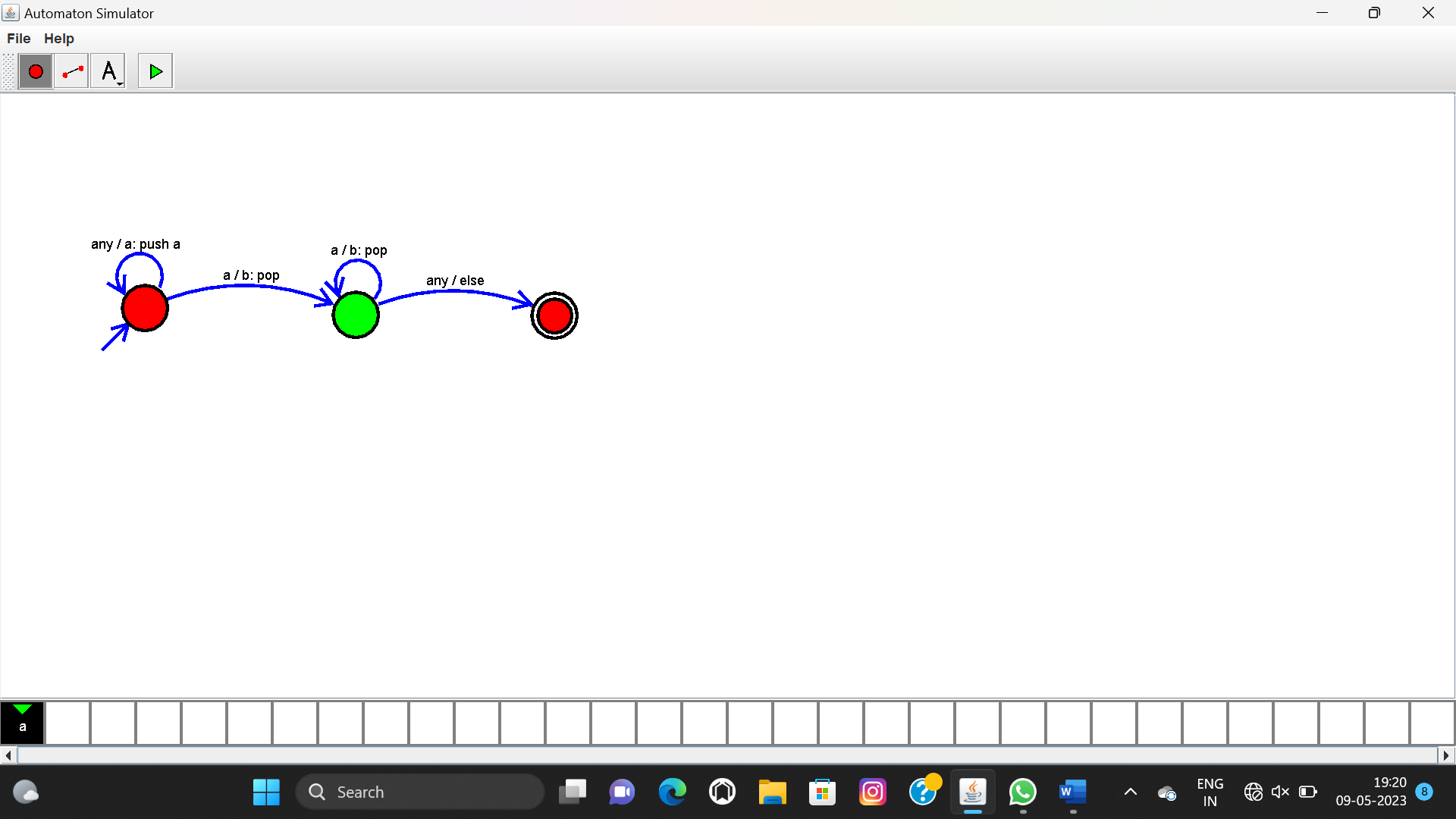
**12.DFA TO ACCPET bcaaaaaaaaaa,bc,and c**



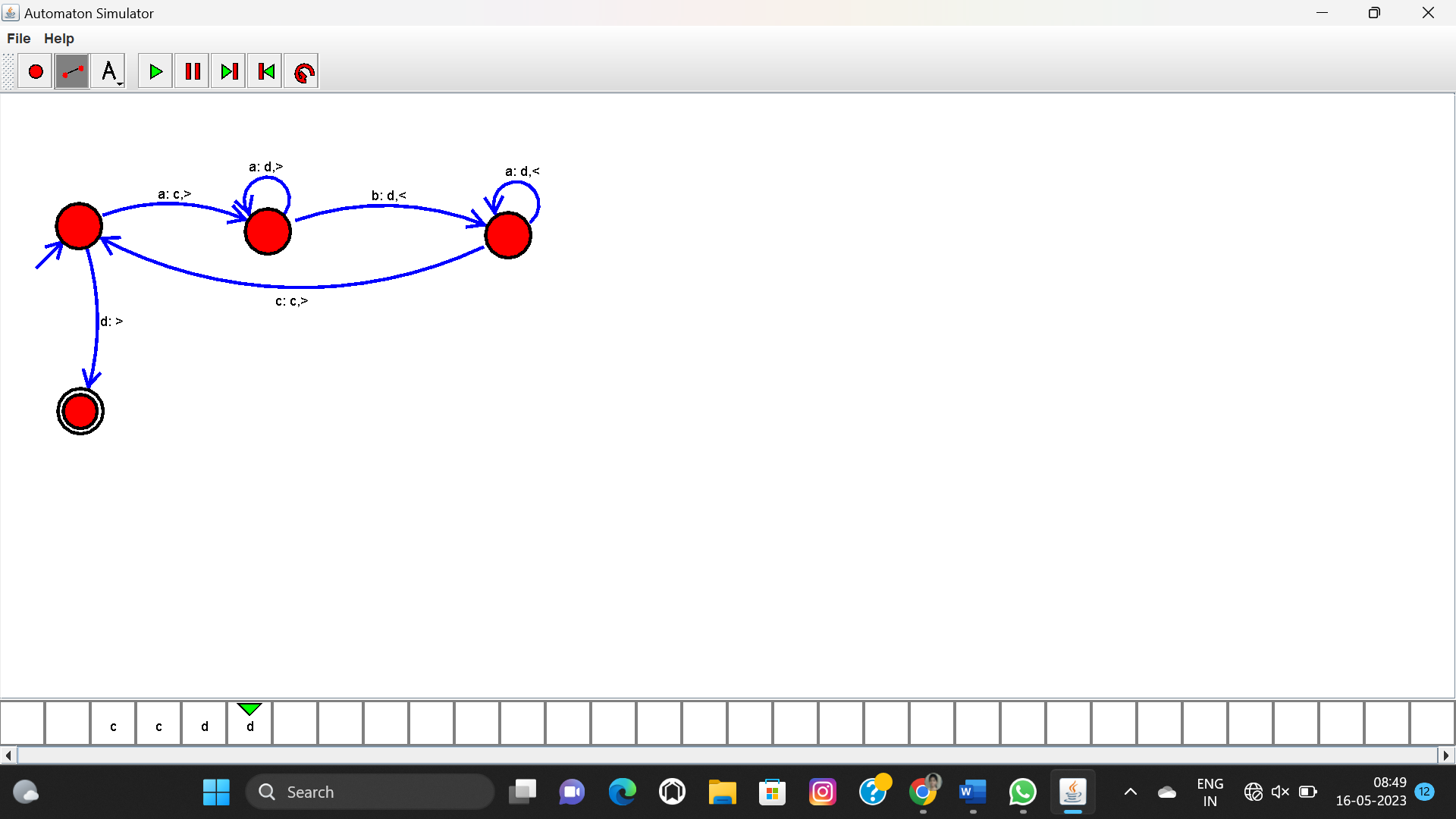
**13.NFA TO ACCEPT aaaaaa**



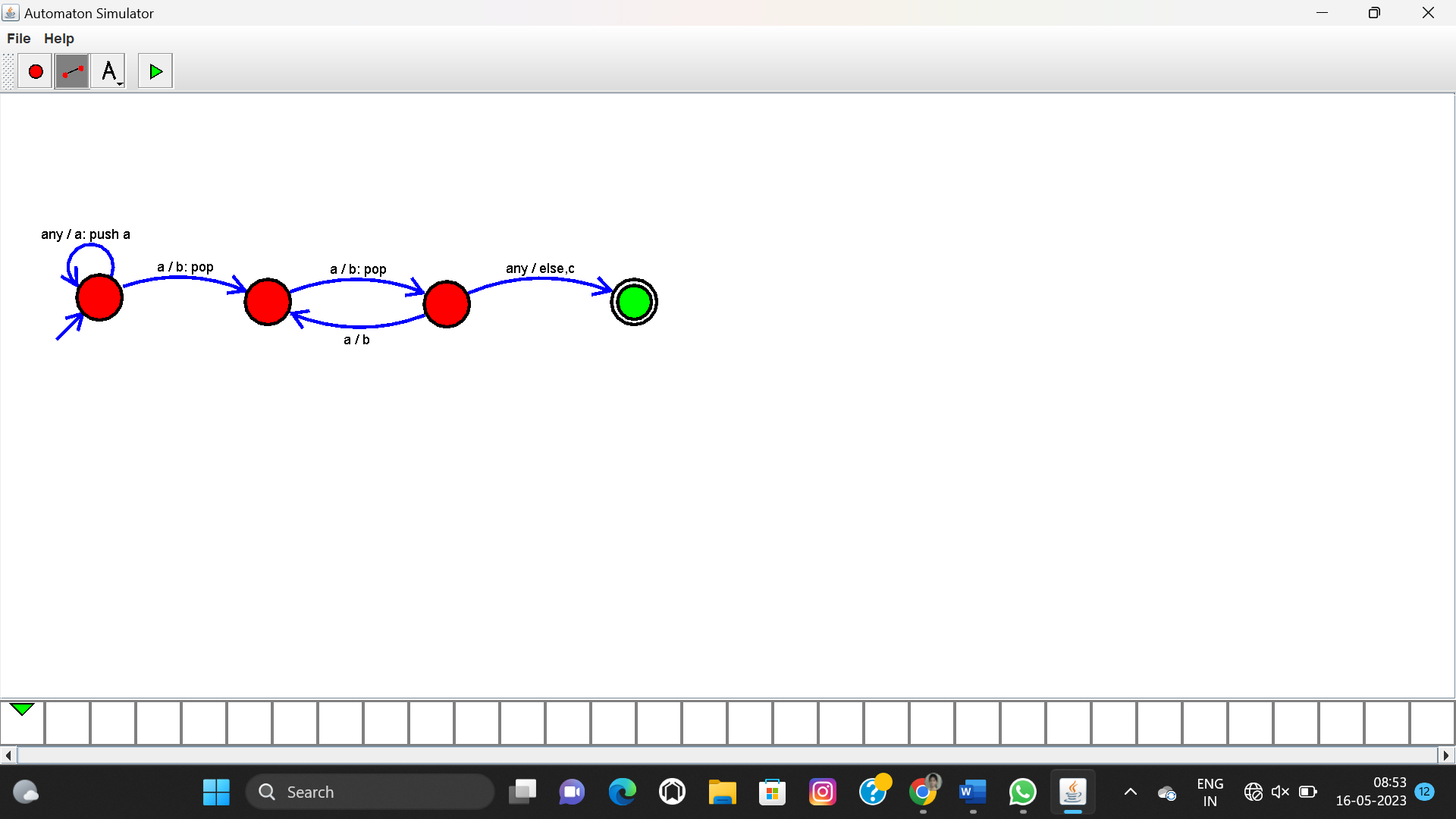
**14.PDA FOR THE INPUT (a^nb^n)**



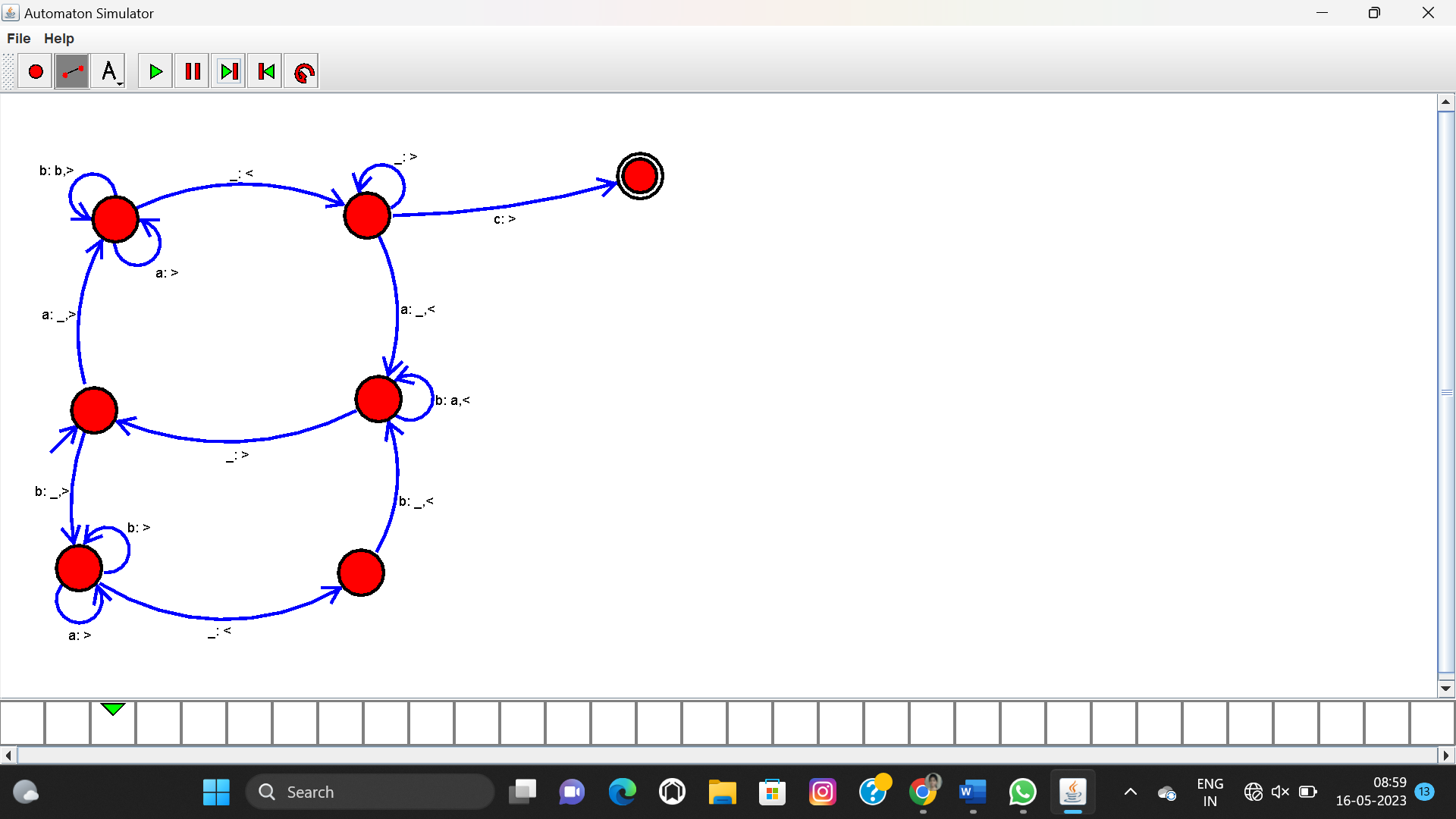
**15.TM FOR INPUT(a^nb^n)**



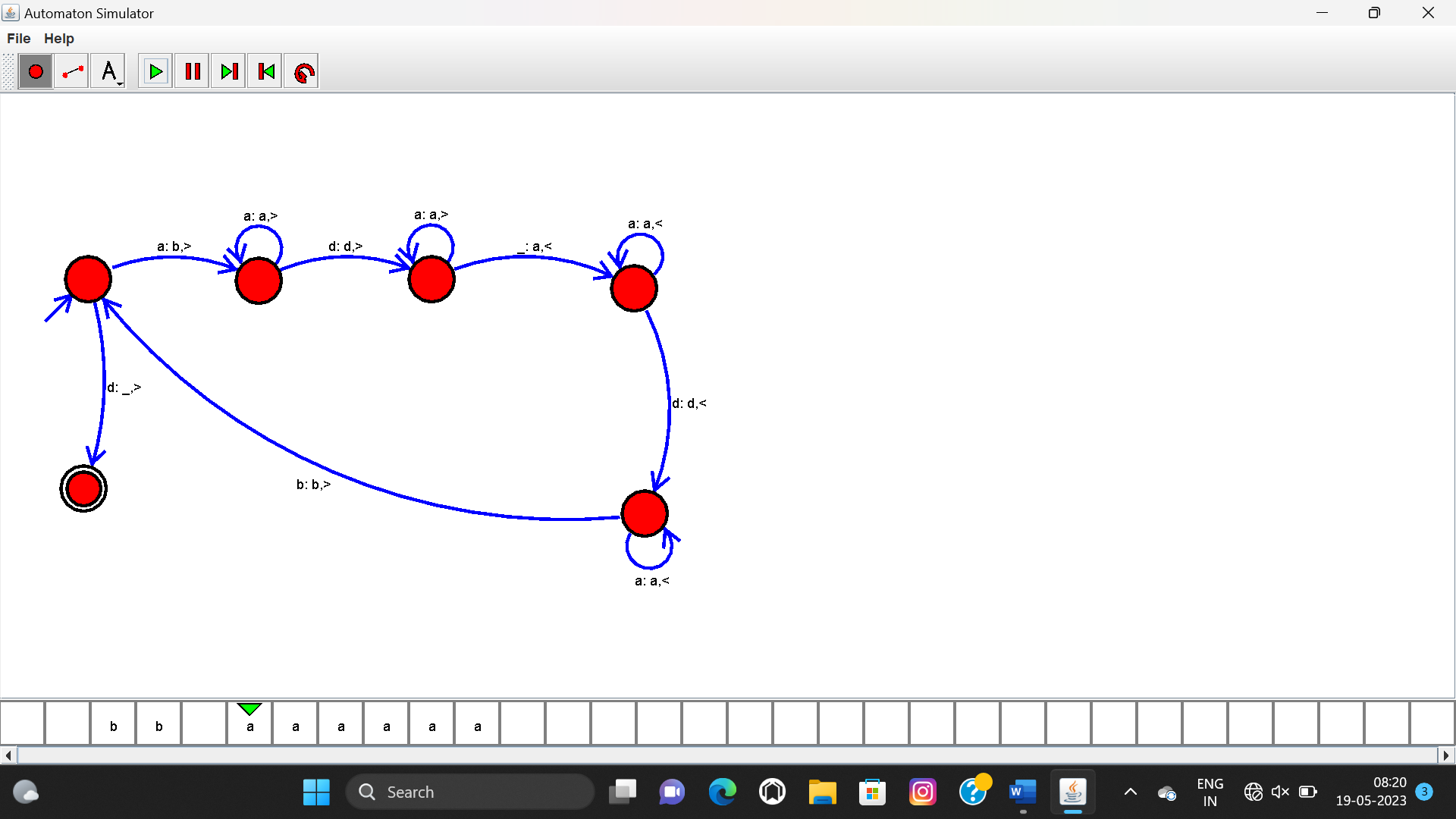
**16.PDA FOR INPUT aabbbbc (L=a^nb^2n)**



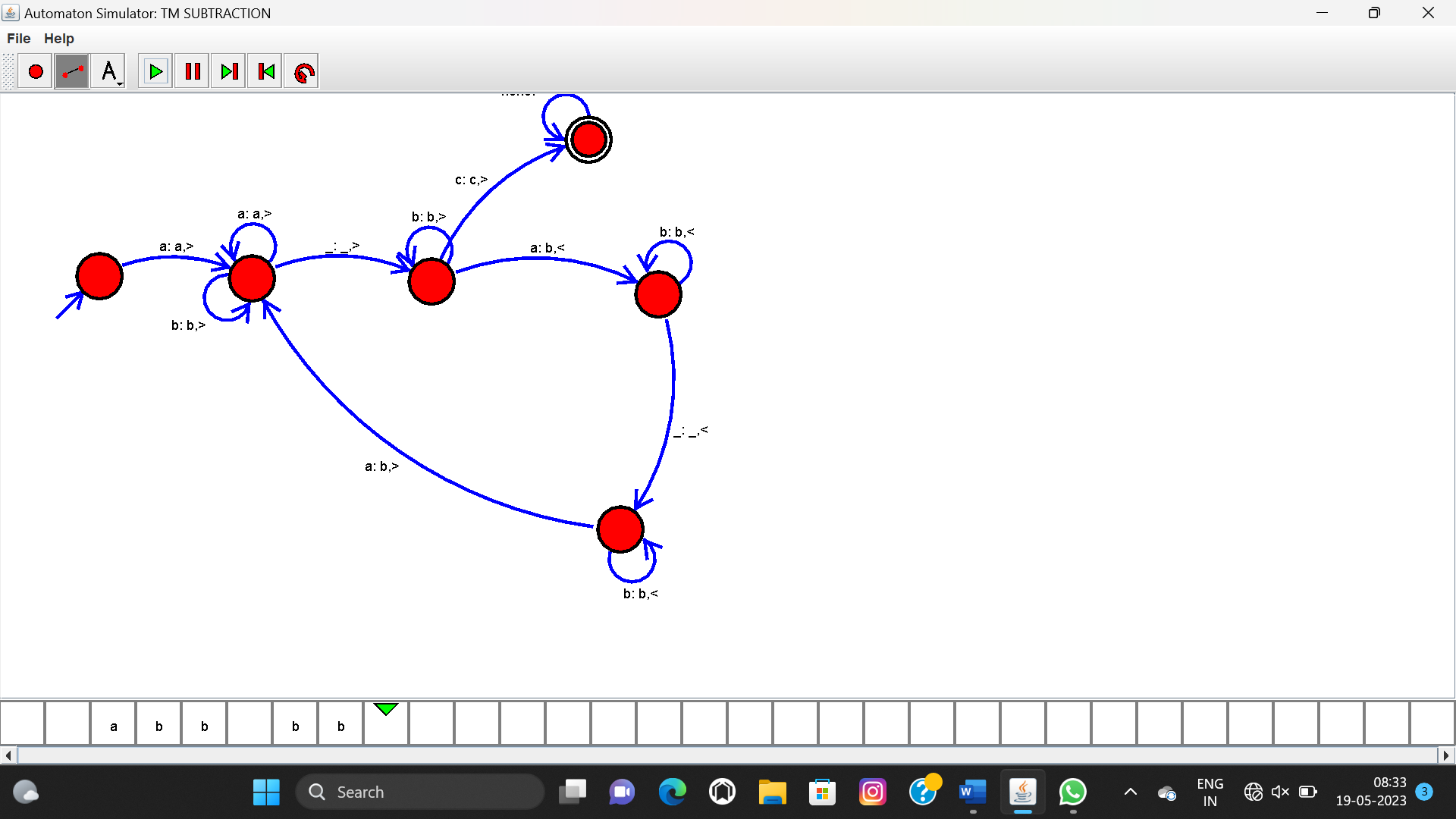
**17.TM SIMULATION FOR PALINDROME W=ababa c**



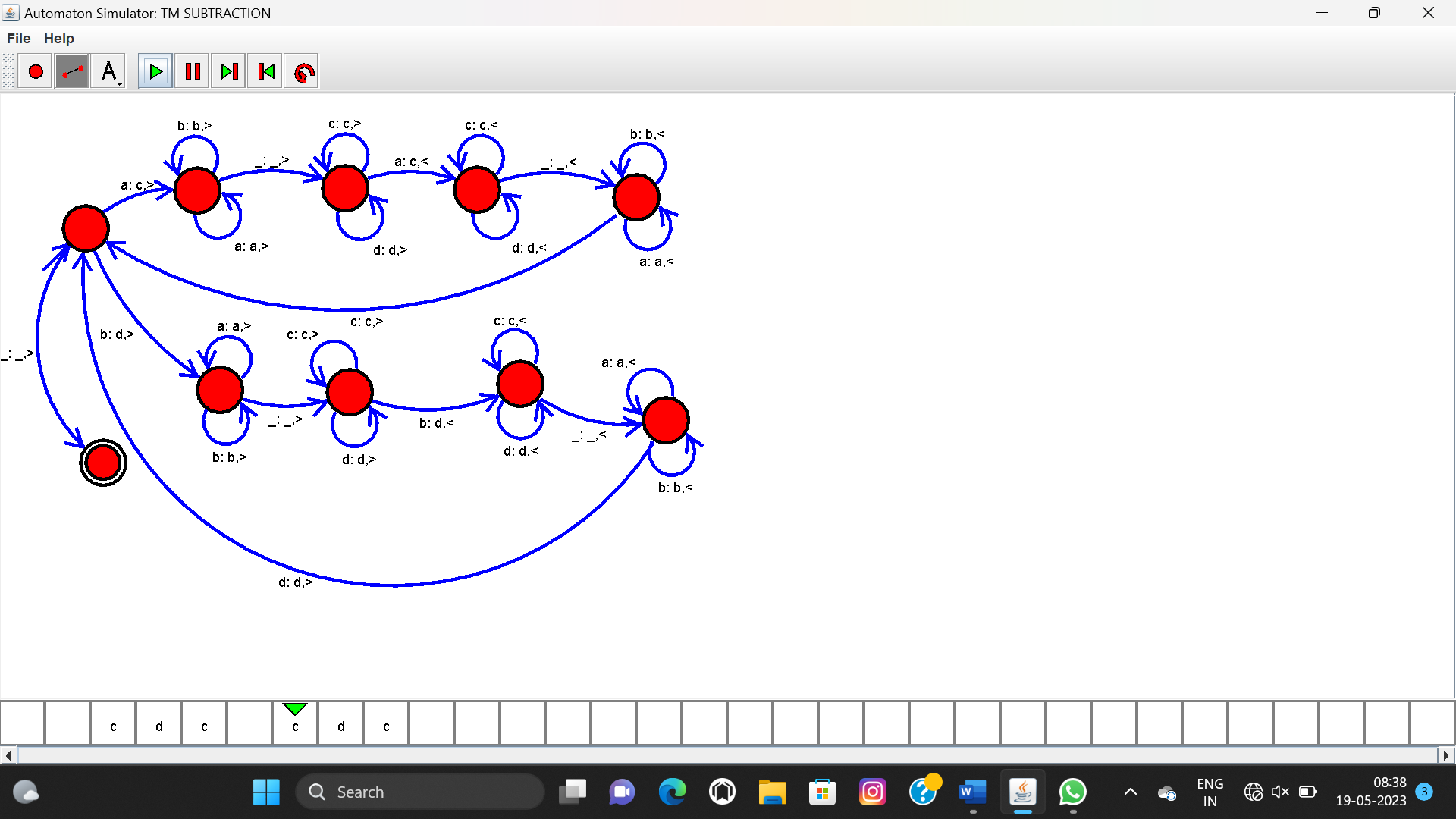
**18.PERFORM TM FOR ADDITION(aa + aaaa)**



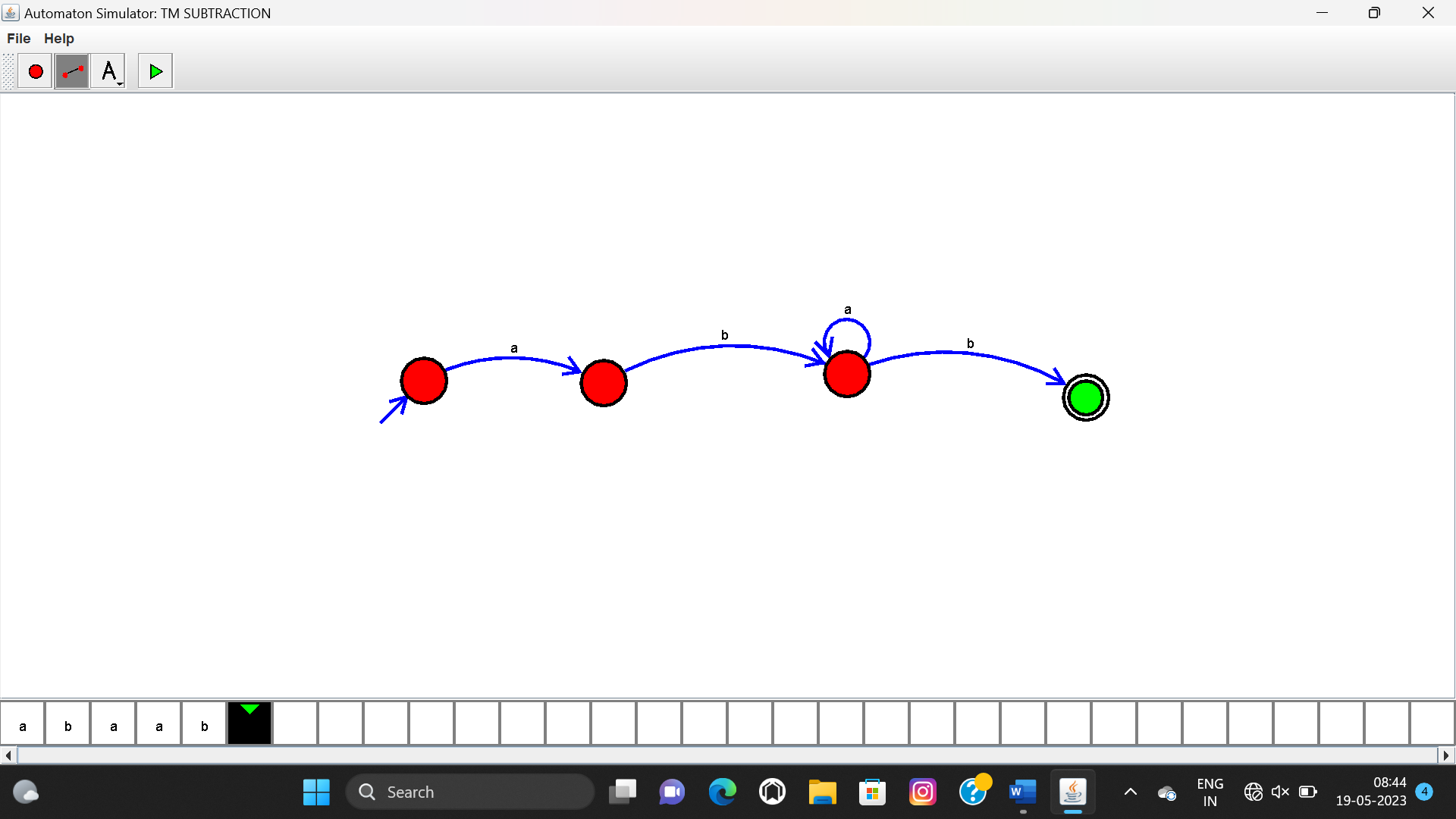
**19.PERFORM TM FOR SUBTRACTION**



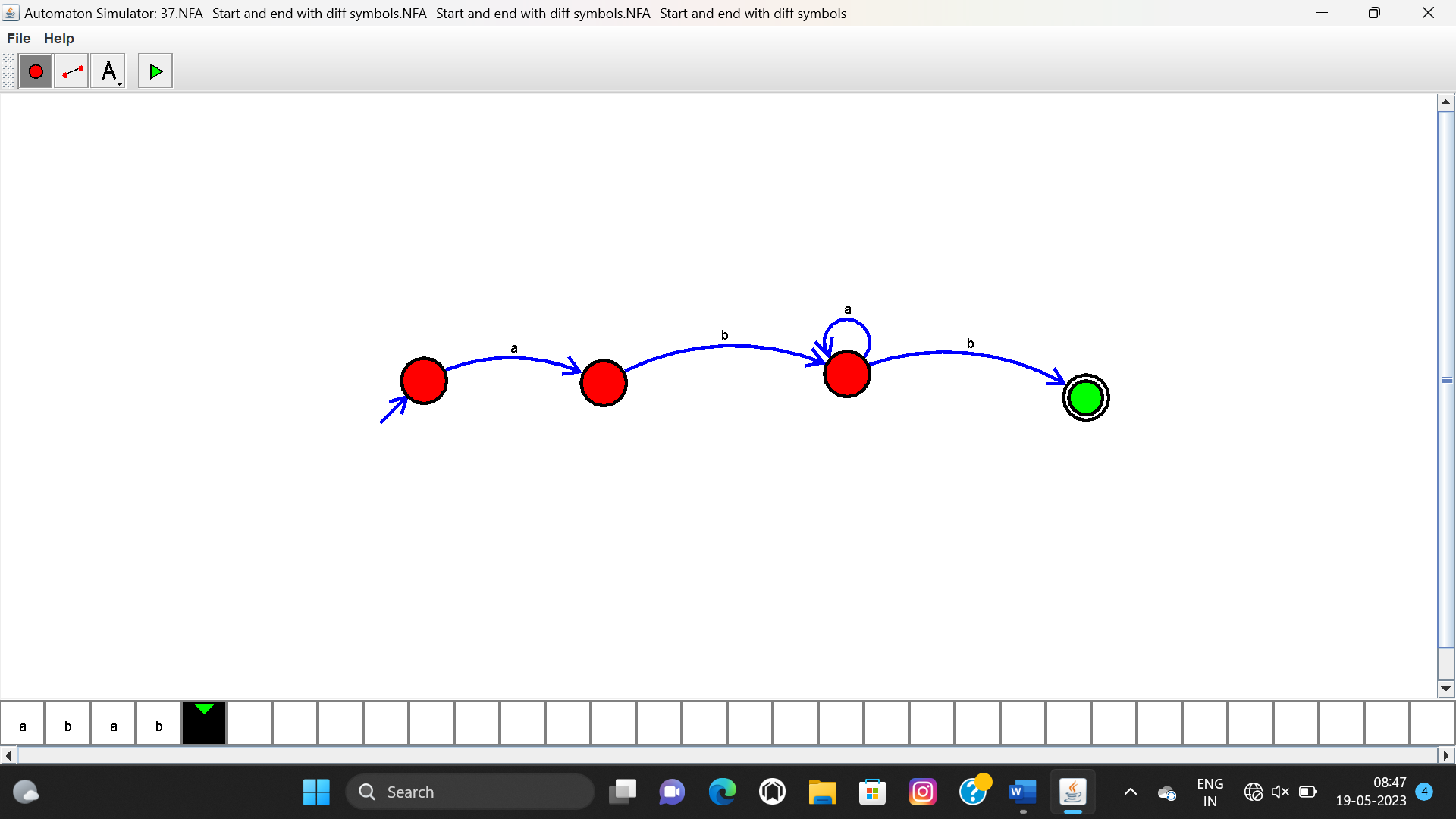
20.DESIGN TM TO PERFORM STRING COMPARISON (W =aba aba)



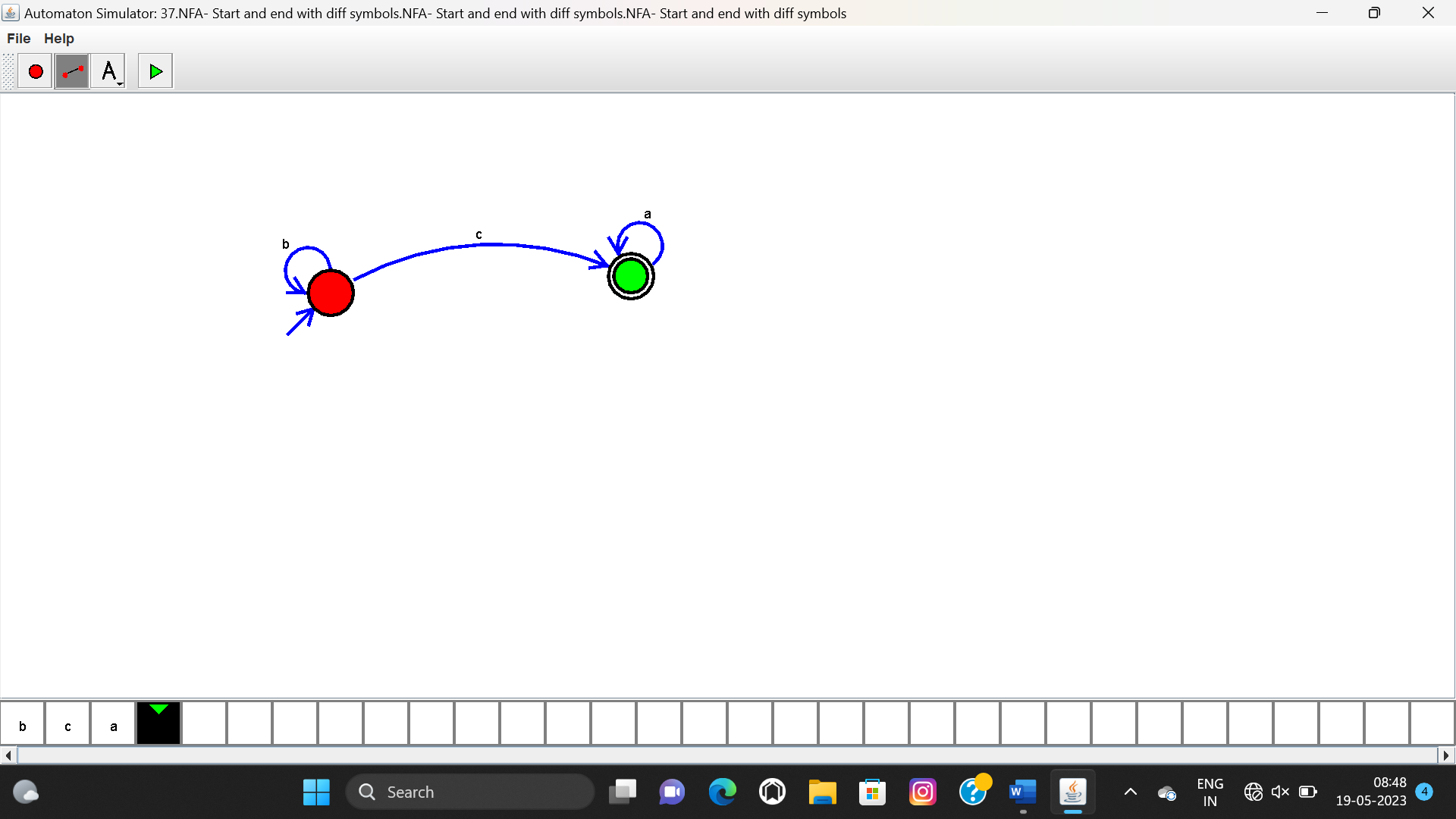
21.NFA (abaab)



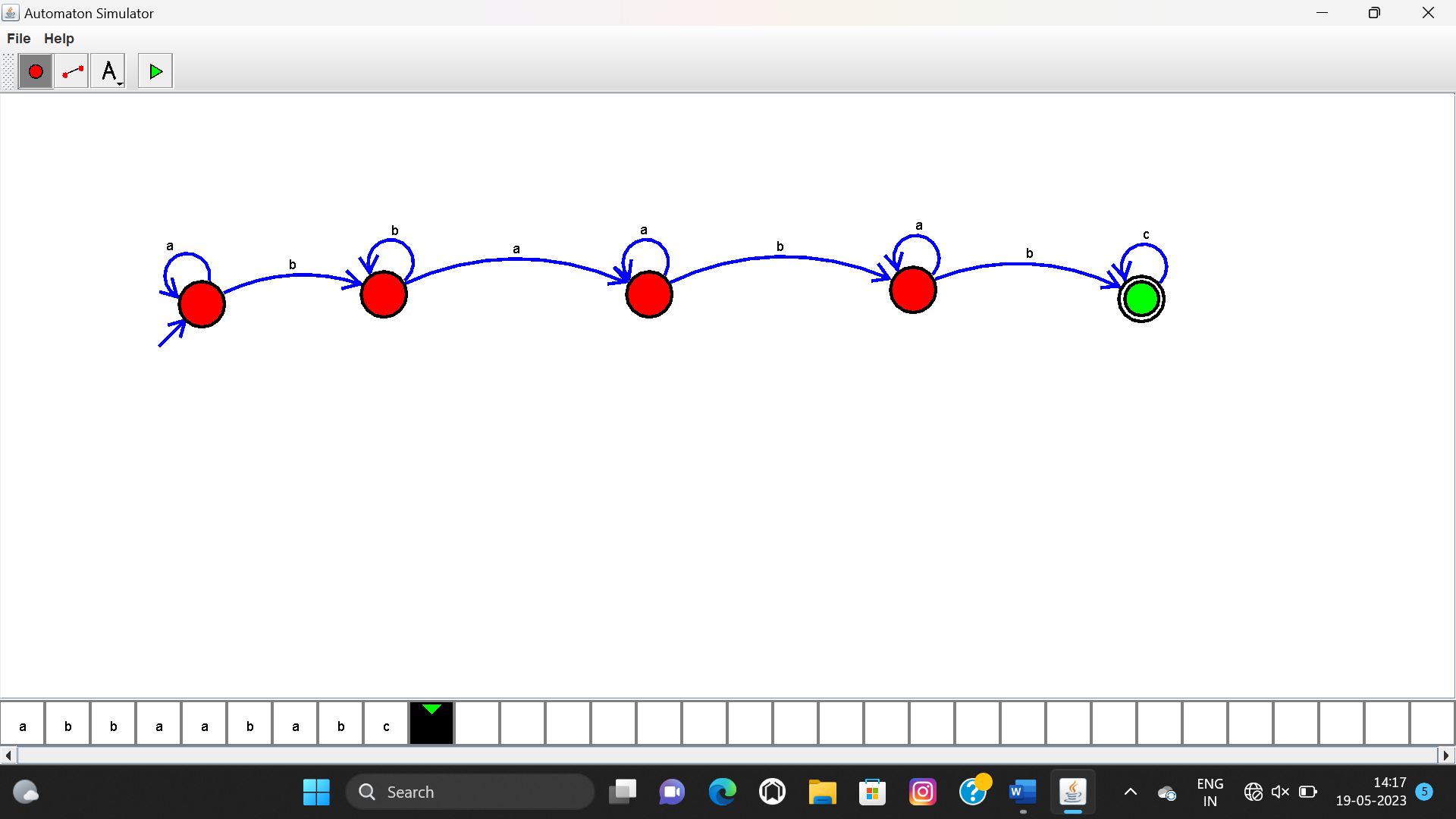
**22.NFA START AND END WITH DIFFERENT SYMBOLS**



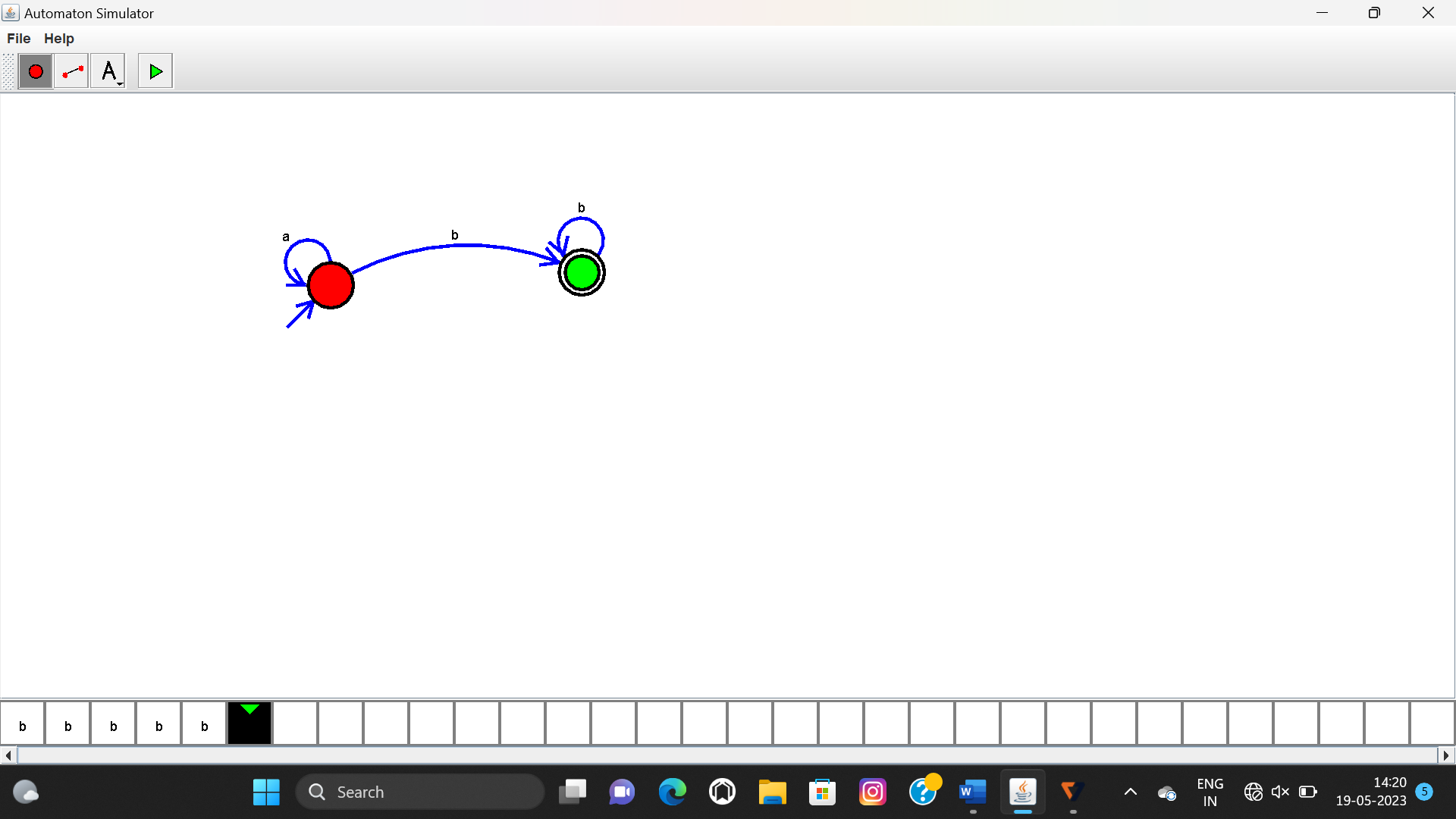
23.NFA (bca)



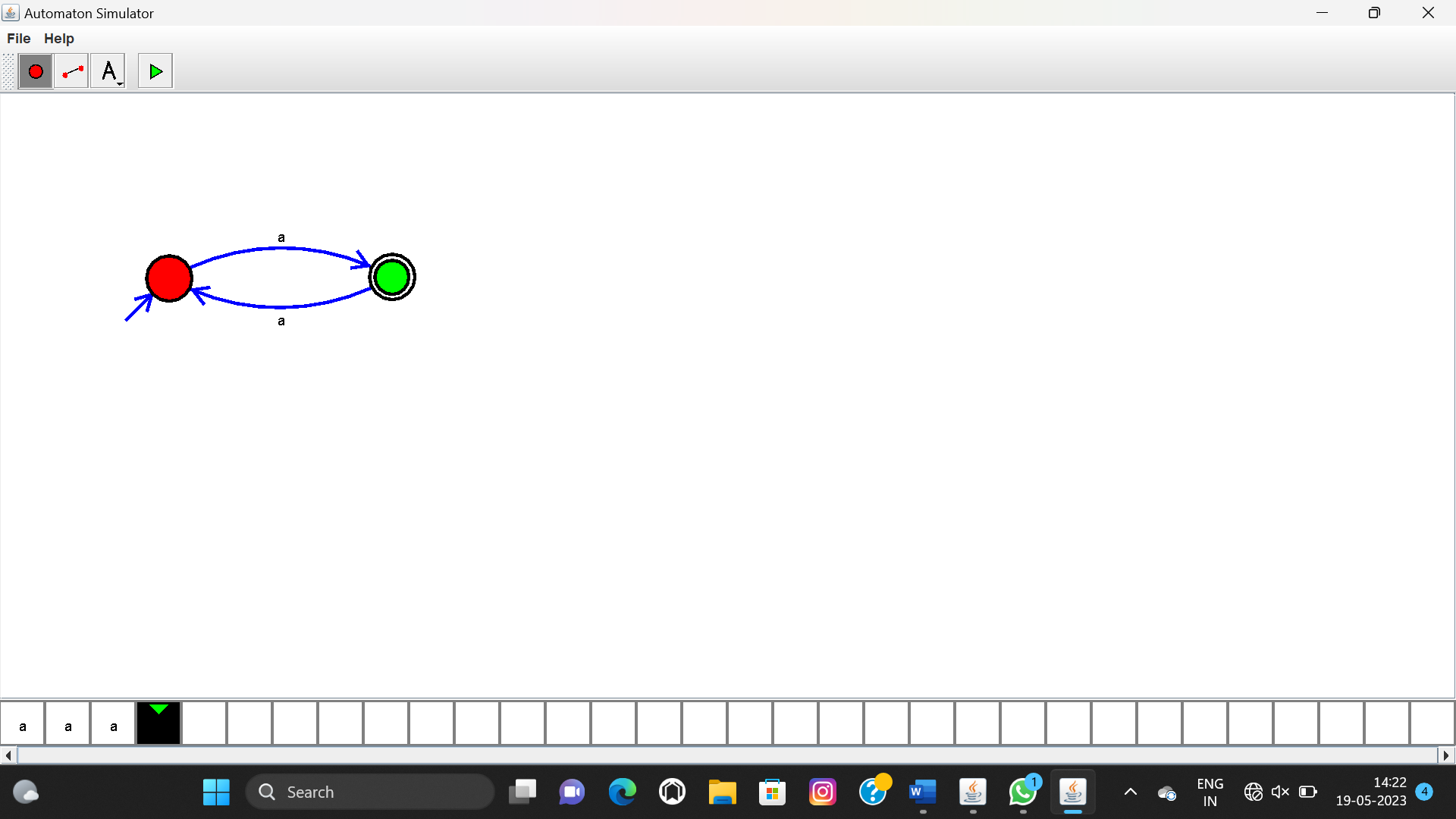
24.DFA(abbaababc)



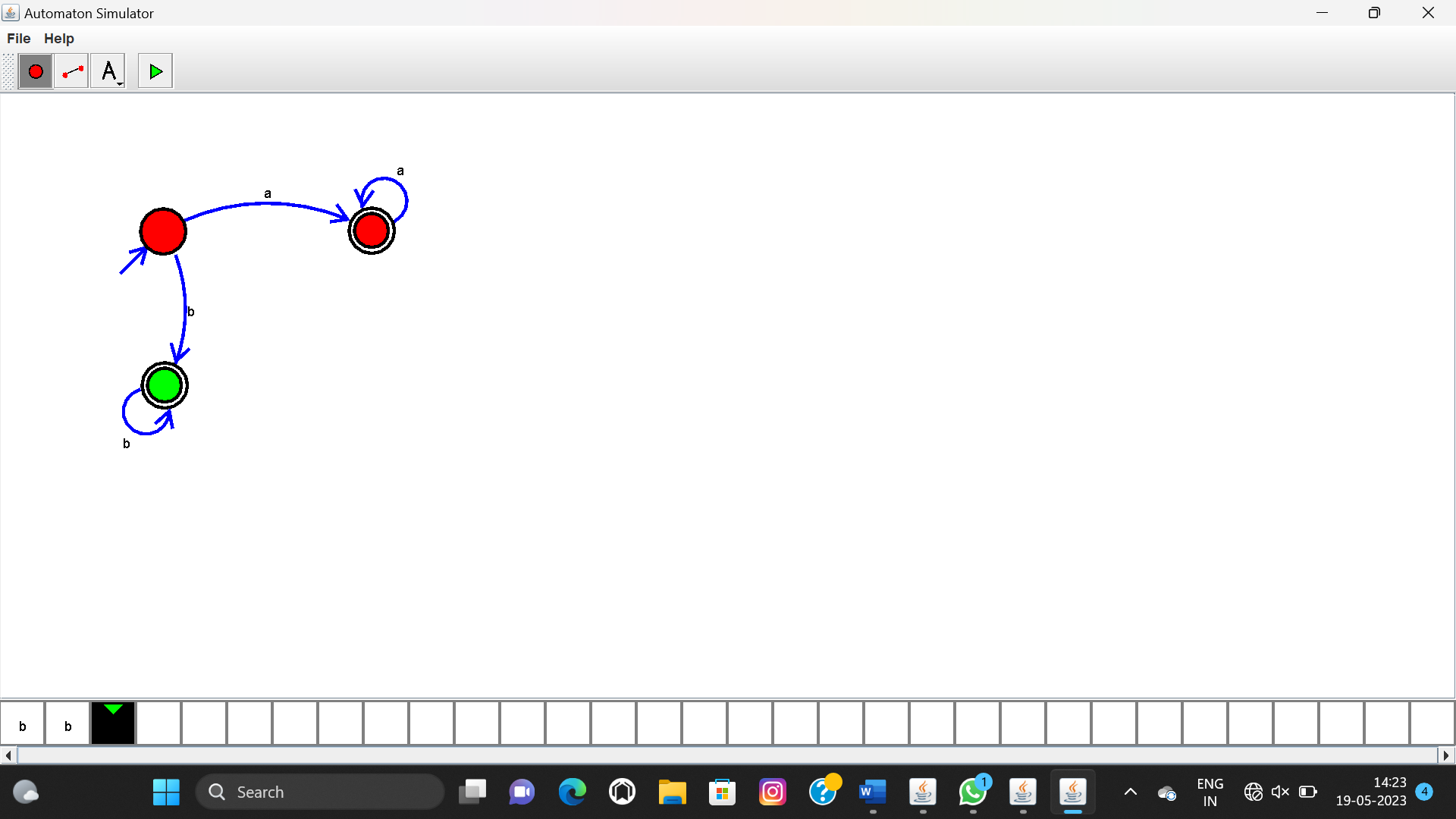
**25.NFA n no of b’s**



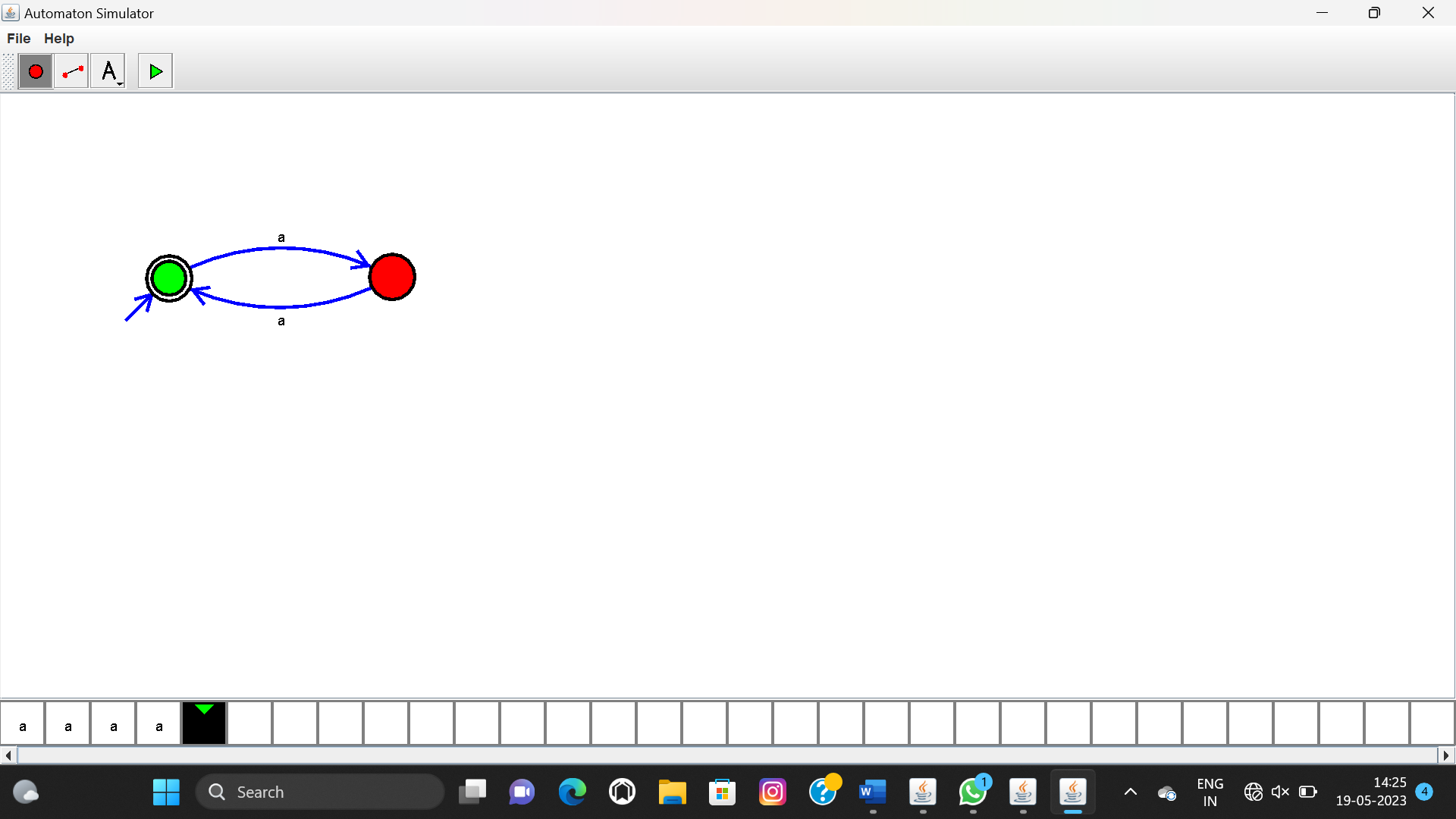
**26.DFA odd a**



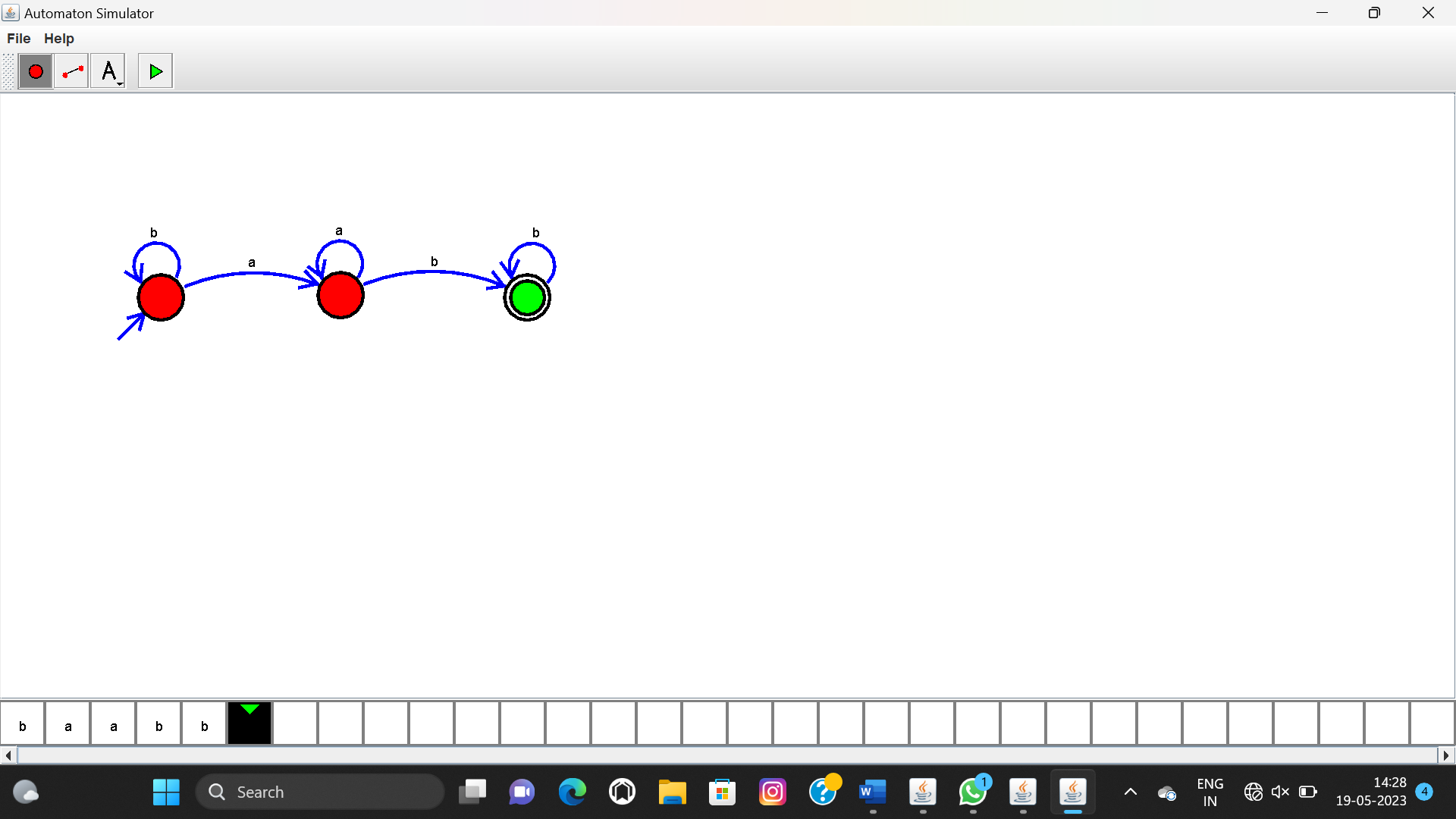
**27.DFA start with a or b**



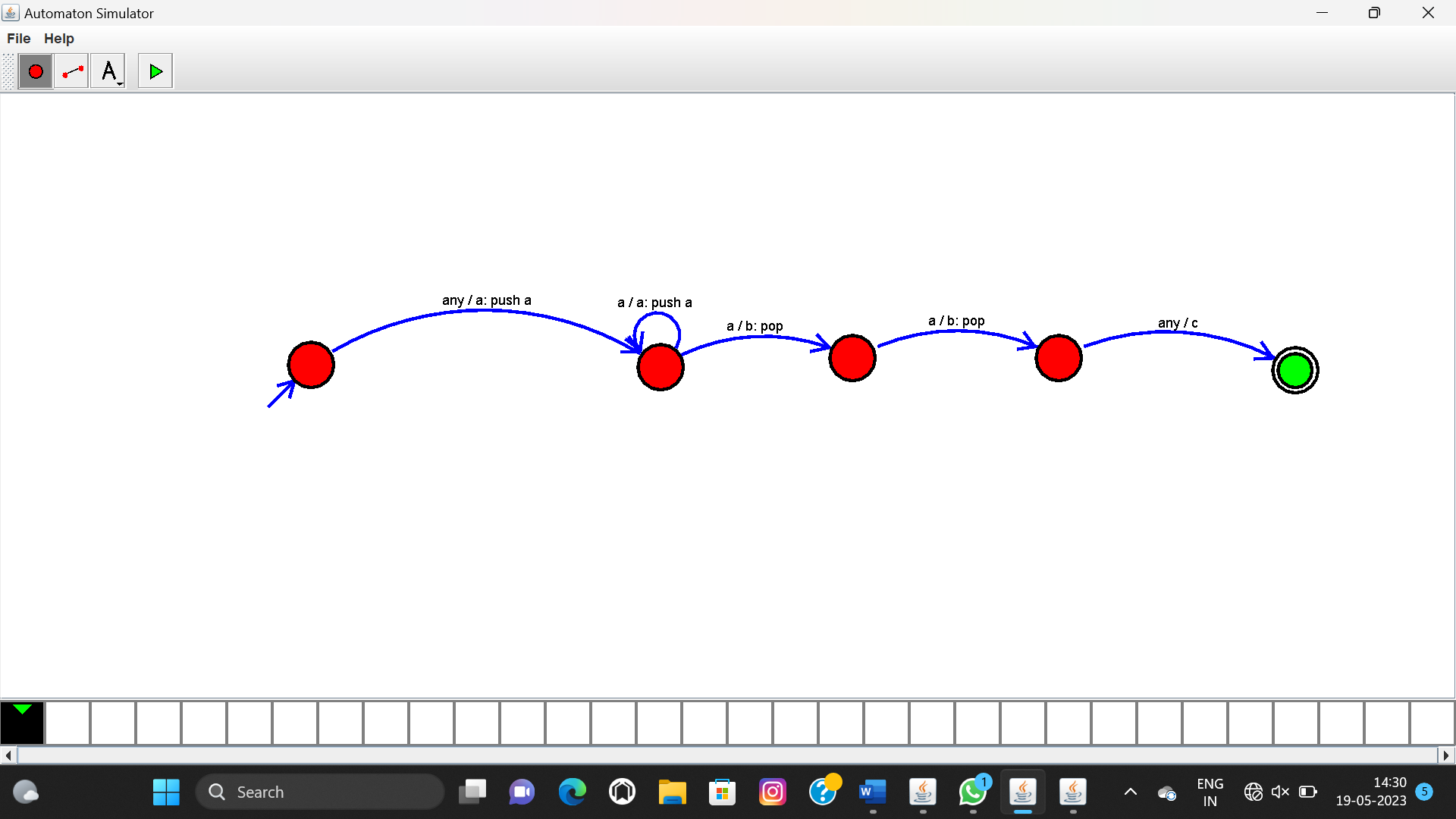
**28.DFA even no of a’s**



**29.DFA substring ab**



**30.PDA-aabbc**



31.