

1. Write a c program to simulate a deterministic finite automata (DFA) for the given language representing strings that starts with a and end with a.

Aim:

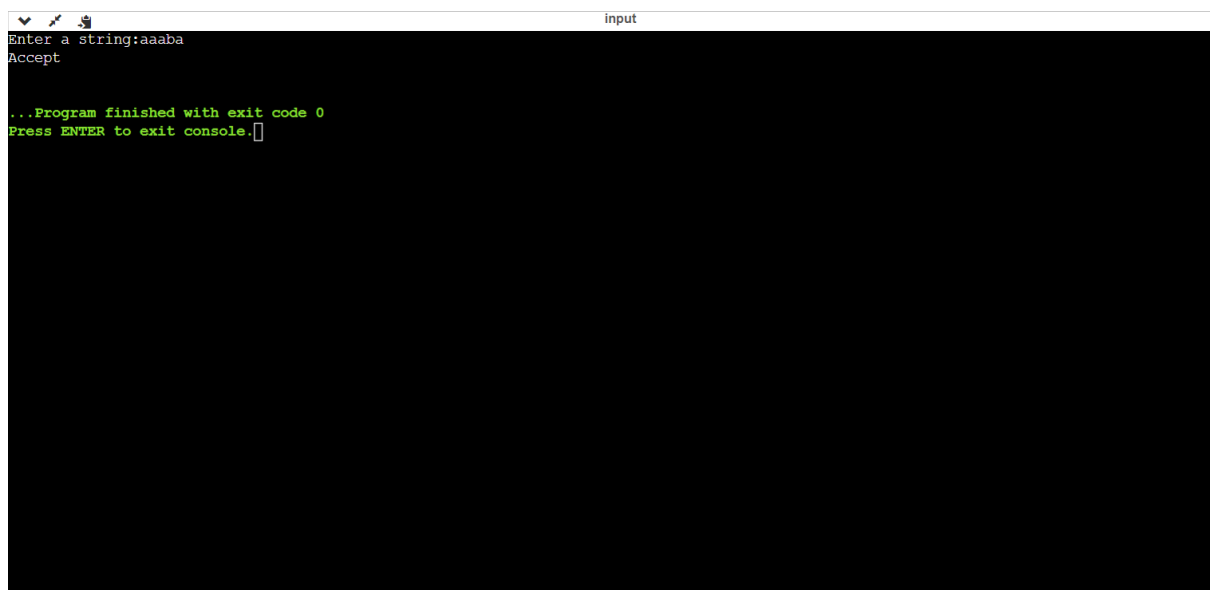
To write a c program to simulate a deterministic finite automata (DFA) for the given language representing strings that starts with a and end with a.

Program:

```
#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("dontAccept\n");
else
printf("Accept\n");
}
```

Output:

A screenshot of a terminal window with a black background and white text. The window title is "input". The text inside the terminal shows the program's execution: it prompts "Enter a string:aaaba", outputs "Accept", and then displays a green message "...Program finished with exit code 0" followed by "Press ENTER to exit console." in white.

```
input
Enter a string:aaaba
Accept

...Program finished with exit code 0
Press ENTER to exit console.
```

2. Write a c program to simulate a deterministic finite automata (DFA) for the given language representing strings that starts with 0 and end with 1.

Aim:

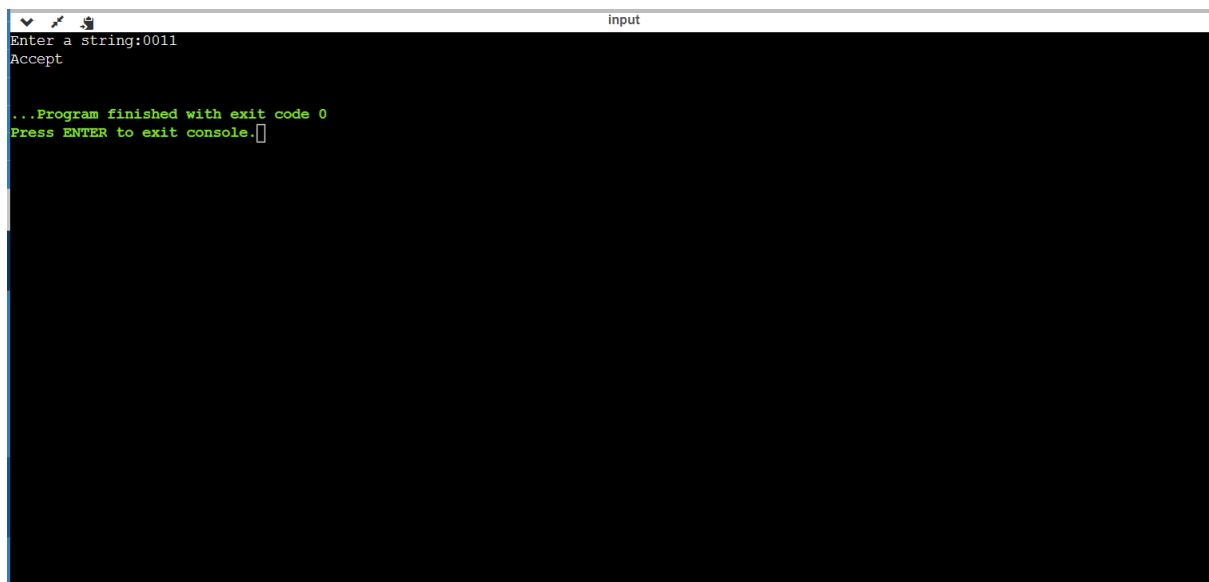
To write a c program to simulate a deterministic finite automata (DFA) for the given language representing strings that starts with 0 and end with 1.

Program:

```
#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=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:

A screenshot of a terminal window titled "input". The terminal has a black background with white text. The first line shows the prompt "Enter a String:0011" followed by the input "Accept". The second line shows the output "Accept". The third line shows the message "...Program finished with exit code 0" in green. The fourth line shows the message "Press ENTER to exit console." in green, followed by a cursor. The terminal window has a standard macOS-style title bar with a red, yellow, and green button on the left and a title bar on the right.

```
input
Enter a String:0011
Accept
...Program finished with exit code 0
Press ENTER to exit console.
```

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

$$S \rightarrow 0A1 \quad A \rightarrow 0A \mid 1A \mid \varepsilon$$

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

int S(char []);
int A(char []);

int main() {
    char str[100];
    printf("Enter a string to check: ");
    scanf("%s", str);

    if(S(str)) {
        printf("String is accepted by the grammar.\n");
    } else {
        printf("String is not accepted by the grammar.\n");
    }

    return 0;
}

int S(char str[]) {
    if(str[0] == '0' && str[strlen(str)-1] == '1') {
        return A(&str[1]);
    } else {
        return 0;
    }
}

int A(char str[]) {
    if(strlen(str) == 0) {
        return 1;
    } else if(str[0] == '0') {
        return A(&str[1]);
    } else if(str[0] == '1') {
        return A(&str[1]);
    } else {
        return 0;
    }
}
```

}

OUTPUT:



```
Enter a string to check: 00100101
String is accepted by the grammar.

...Program finished with exit code 0
Press ENTER to exit console.[]
```

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

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$$

PROGRAM:

```
#include <stdio.h>

#include <string.h>

int main() {

    char input[100];

    int length, i;

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

    scanf("%s", input);

    length = strlen(input);

    // Check if the input string is valid
    for (i = 0; i < length; i++) {
        if (input[i] != '0' && input[i] != '1') {
            printf("Invalid string: contains characters other than 0 and 1\n");
            return 0;
        }
    }
}
```

```


if (length == 0 || input[0] == '0' || input[0] == '1') {
    for (i = 1; i < length; i++) {
        if (input[i] != '0' && input[i] != '1') {
            printf("Invalid string: contains characters other than 0 and 1\n");
            return 0;
        }
    }
    printf("The string is accepted\n");
} else if (input[0] == '0' && input[length - 1] == '0') {
    char substring[100];
    strncpy(substring, input + 1, length - 2);
    substring[length - 2] = '\0';
    if (strlen(substring) == 0 || (strlen(substring) == 1 && (substring[0] == '0' || substring[0] ==
'1')))) {
        printf("The string is accepted\n");
    } else {
        printf("The string is not accepted\n");
    }
} else if (input[0] == '1' && input[length - 1] == '1') {
    char substring[100];
    strncpy(substring, input + 1, length - 2);
    substring[length - 2] = '\0';
    if (strlen(substring) == 0 || (strlen(substring) == 1 && (substring[0] == '0' || substring[0] ==
'1')))) {
        printf("The string is accepted\n");
    } else {
        printf("The string is not accepted\n");
    }
} else {
    printf("The string is not accepted\n");
}

```



```
}  
  
return 0;  
}
```

OUTPUT:



The screenshot shows a terminal window with a title bar containing a dropdown arrow, a pencil icon, and a close icon, followed by the text "input". The terminal content is as follows:

```
Enter a string to check: 00110010  
The string is accepted  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

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

$S \rightarrow 0S0 \mid A$ $A \rightarrow 1A \mid \varepsilon$

PROGRAM:

```
#include <stdio.h>

#include <string.h>

#define MAX_LENGTH 100

int is_valid(char string[], int start, int end);

int S(char string[], int start, int end) {
    if (string[start] == '0' && string[end] == '0' && is_valid(string, start+1, end-1)) {
        return 1;
    } else {
        return 0;
    }
}

int A(char string[], int start, int end) {
    if (start > end) {
        return 1;
    } else if (string[start] == '1' && A(string, start+1, end)) {
        return 1;
    } else {
        return 0;
    }
}

int is_valid(char string[], int start, int end) {
    for (int i = start; i <= end; i++) {
        if (string[i] != '0' && string[i] != '1') {
            return 0;
        }
    }
    return 1;
}
```

```
}  
  
int main() {  
    char input_string[MAX_LENGTH];  
    printf("Enter input string: ");  
    scanf("%s", input_string);  
    int string_length = strlen(input_string);  
    if (S(input_string, 0, string_length - 1)) {  
        printf("Accepted\n");  
    } else {  
        printf("Rejected\n");  
    }  
    return 0;  
}
```

OUTPUT:



The screenshot shows a terminal window titled "input". The user has entered the string "0011100" in response to the prompt "Enter input string: ". The program has printed "Accepted" on the next line. At the bottom of the terminal, there is a green message: "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a cursor.

```
input  
Enter input string: 0011100  
Accepted  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

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

$$S \rightarrow 0S1 \mid \varepsilon$$

PROGRAM:

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

```
#include <stdlib.h>
```

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

```
#define MAX_LENGTH 1000
```

```
int is_valid(char* string);
```

```
int main() {
```

```
    char string[MAX_LENGTH];
```

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

```
    scanf("%s", string);
```

```
    if (is_valid(string)) {
```

```
        printf("The string is in the language defined by the CFG.\n");
```

```
    } else {
```

```
        printf("The string is not in the language defined by the CFG.\n");
```

```
    }
```

```
    return 0;
```

```
}
```

```
int is_valid(char* string) {
```

```
    if (strlen(string) == 0) {
```

```
        return 1; // the empty string is in the language
```

```
    }
```

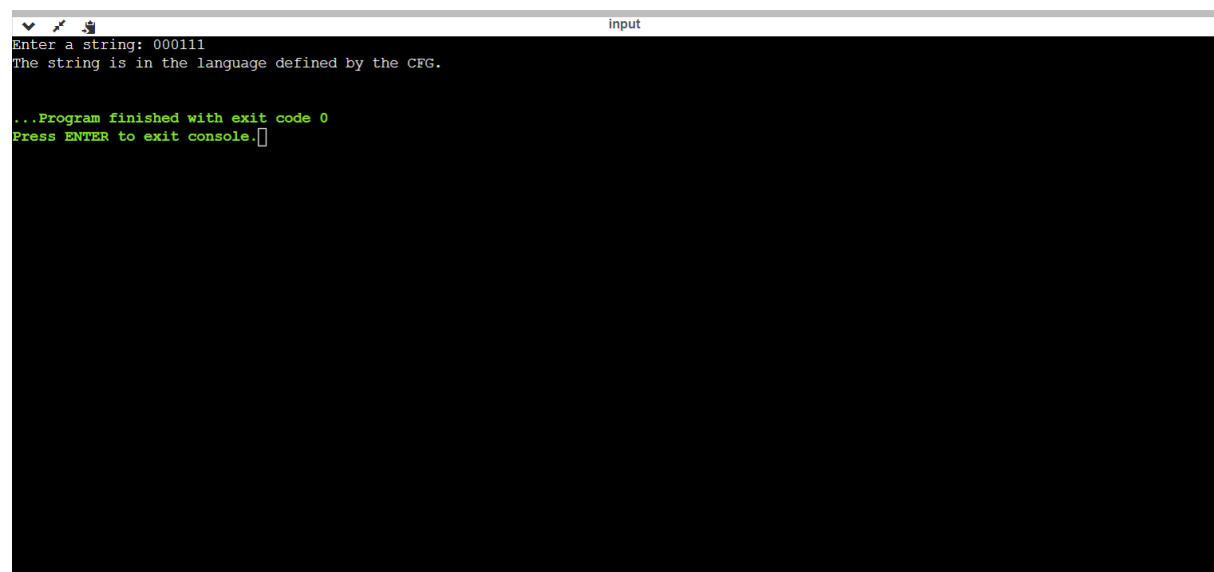
```
    if (string[0] == '0' && string[strlen(string)-1] == '1') {
```

```
        char* new_string = (char*) malloc((strlen(string)-2) * sizeof(char));
```

```
        strncpy(new_string, string+1, strlen(string)-2);
```

```
new_string[strlen(string)-2] = '\0';  
if (is_valid(new_string)) {  
    return 1;  
}  
}  
return 0;  
}
```

OUTPUT:



The screenshot shows a terminal window with a title bar containing a window icon, a maximize icon, and a close icon, followed by the text "input". The terminal content is as follows:

```
Enter a String: 000111  
The string is in the language defined by the CFG.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

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

$$S \rightarrow A101A, \quad A \rightarrow 0A \mid 1A \mid \epsilon$$

PROGRAM:

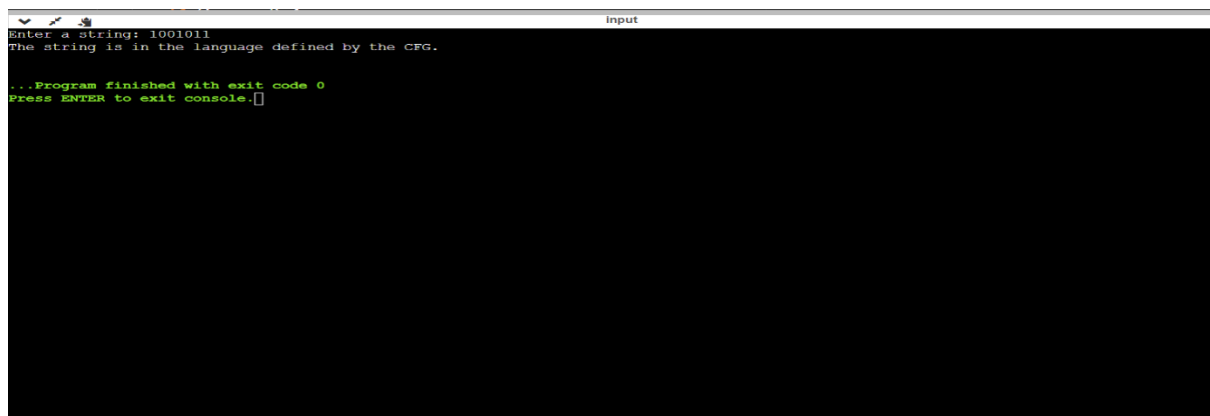
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LENGTH 1000
int is_valid(char* string, int start, int end);
int main() {
    char string[MAX_LENGTH];
    printf("Enter a string: ");
    scanf("%s", string);
    if (is_valid(string, 0, strlen(string)-1)) {
        printf("The string is in the language defined by the CFG.\n");
    } else {
        printf("The string is not in the language defined by the CFG.\n");
    }
    return 0;
}
int is_valid(char* string, int start, int end) {
    if (start > end) {
        return 1; // the empty string is in the language
    }
    if (string[start] == '1' && string[end] == '1' && end == start + 2) {
        return 1; // S -> A101A
    }
    if (string[start] == '0' || string[start] == '1') {
        for (int i = start+1; i <= end; i++) {
            if (string[i] != '0' && string[i] != '1') {
                break;
            }
        }
    }
}
```

```

    }
    if (i == end) {
        return 1; // A -> 0A | 1A | e
    }
}
for (int i = end-1; i >= start; i--) {
    if (string[i] != '0' && string[i] != '1') {
        break;
    }
    if (i == start) {
        return 1; // A -> 0A | 1A | e
    }
}
for (int i = start+1; i <= end-1; i++) {
    if (string[i] == '0' || string[i] == '1') {
        if (is_valid(string, start+1, i-1) && is_valid(string, i+1, end)) {
            return 1; // A -> 0A | 1A | e
        }
    }
}
}
return 0;}

```

OUTPUT:



The screenshot shows a console window with a dark background. At the top, there is a title bar with a small icon and the word "input". The main area of the console displays the following text: "Enter a String: 1001011", "The string is in the language defined by the CFG.", "...Program finished with exit code 0", and "Press ENTER to exit console." followed by a cursor icon.

```

input
Enter a String: 1001011
The string is in the language defined by the CFG.
...Program finished with exit code 0
Press ENTER to exit console.

```

8. 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

PROGRAM:

```
#include <stdio.h>

#include <string.h>

#define MAX_LENGTH 100

int nfa[MAX_LENGTH][2];

int start_state = 0;

int accept_state = 1;

int simulate_nfa(char* input_string) {
    int input_length = strlen(input_string);
    int current_state = start_state;
    for (int i = 0; i < input_length; i++) {
        int symbol = (input_string[i] == 'a') ? 1 : 0;
        int next_state = nfa[current_state][symbol];
        if (next_state == -1) {
            return 0;
        }
        current_state = next_state;
    }
    return current_state == accept_state;
}

int main() {
    nfa[0][0] = -1; // Reject b as the first symbol
    nfa[0][1] = 1;
    nfa[1][0] = 1;
```



```
nfa[1][1] = 1;

char input[MAX_LENGTH];

printf("Enter an input string: ");

scanf("%s", input);

printf("Input: %s\nResult: %s\n", input, simulate_nfa(input) ? "Accepted" : "Rejected");

return 0;
}
```

OUTPUT:



```
input
Enter an input string: aaababab
Input: aaababab
Result: Accepted

...Program finished with exit code 0
Press ENTER to exit console.
```

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

PROGRAM:

```
#include <stdio.h>

#include <string.h>

#define MAX_LENGTH 100
int main() {
    char input_string[MAX_LENGTH];
    printf("Enter input string: ");
    scanf("%s", input_string);
    int string_length = strlen(input_string);
    if (string_length > 0 && input_string[0] == '0' && input_string[string_length - 1] == '1') {
        printf("Accepted\n");
    } else {
        printf("Rejected\n");
    }
    return 0;
}
```

OUTPUT:



```
input
Enter input string: 0010101
Accepted

...Program finished with exit code 0
Press ENTER to exit console.
```

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

PROGRAM:

```
#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;

} } }

num_states=3;

num_symbols=2;

symbol[10]='e';

n=1;

trans_table[0][0][0]=1;

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;

```

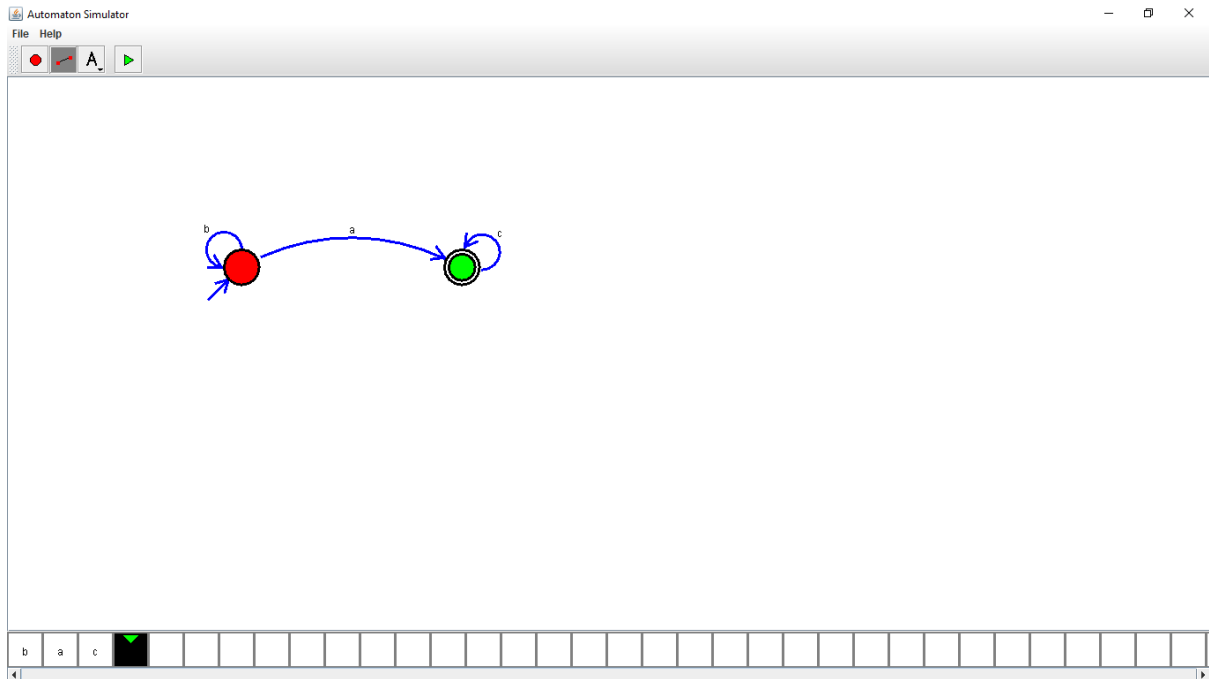
```
}  
num_trans=i;  
for(j=0;j<num_trans;j++)  
{  
e_closure[state][ptr]=y[j];  
ptr++;  
find_e_closure(y[j]);  
}}
```

OUTPUT:

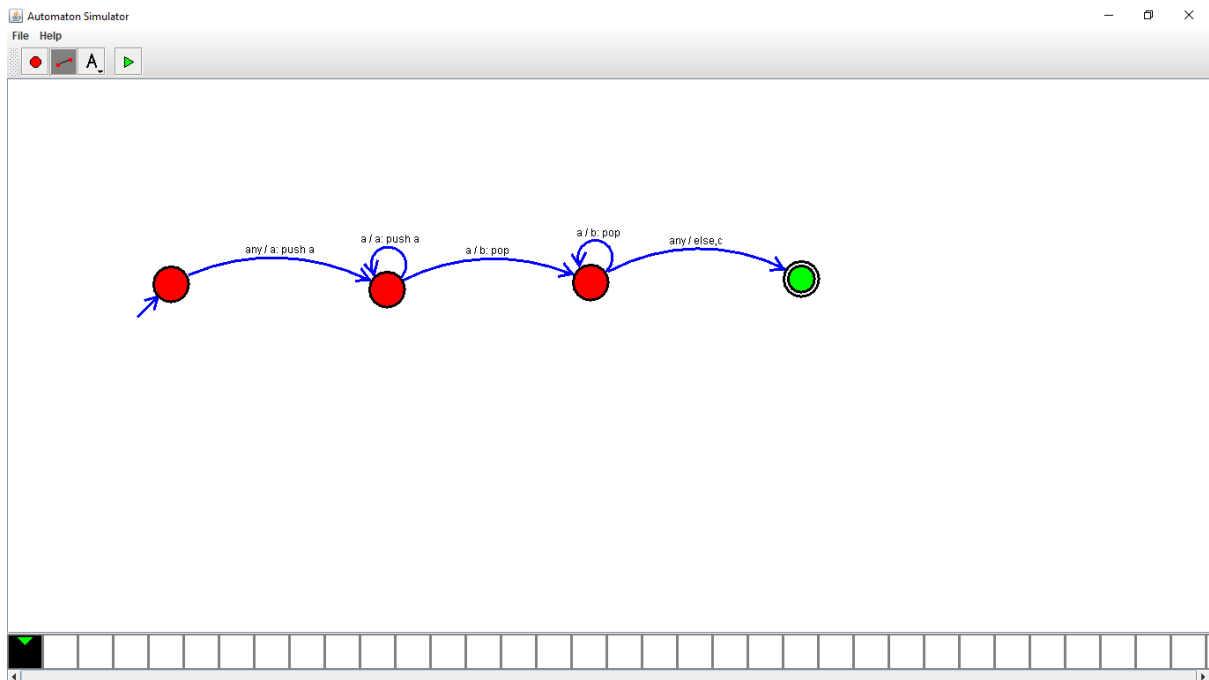
A screenshot of a terminal window with a dark background. The title bar at the top shows a window icon, a maximize icon, and a close icon, followed by the text "input". The terminal content shows the output of a program: "e-closure(0)= {0, 1, }", "e-closure(1)= {1, }", and "e-closure(2)= {2, }". Below these, it says "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a cursor icon.

```
input  
e-closure(0)= {0, 1, }  
e-closure(1)= {1, }  
e-closure(2)= {2, }  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

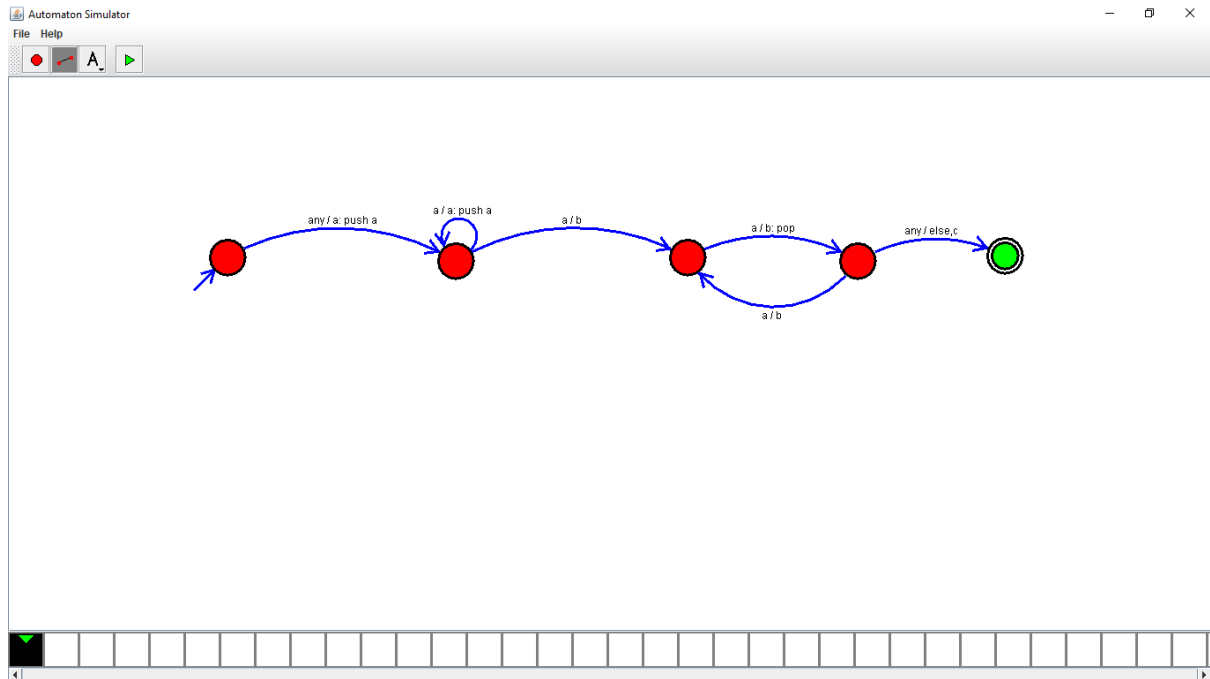
12.Design DFA using simulator to accept the input string "a" ,"ac",and "bac".



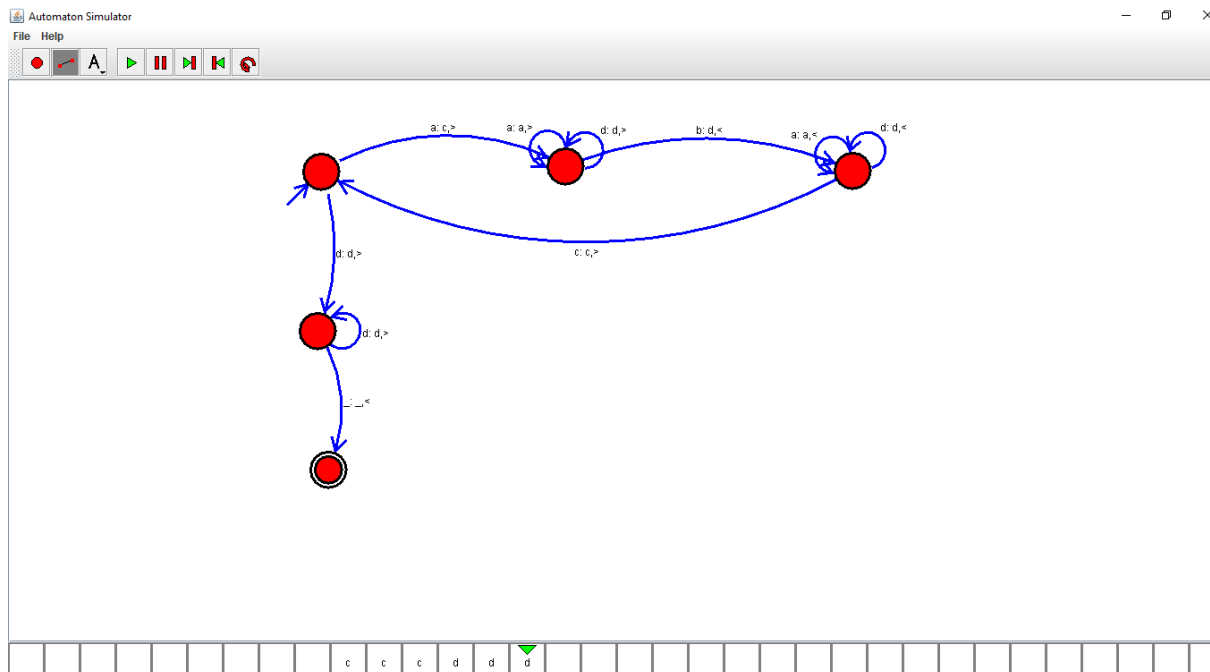
13.Design PDA using simulator to accept the input string aabb



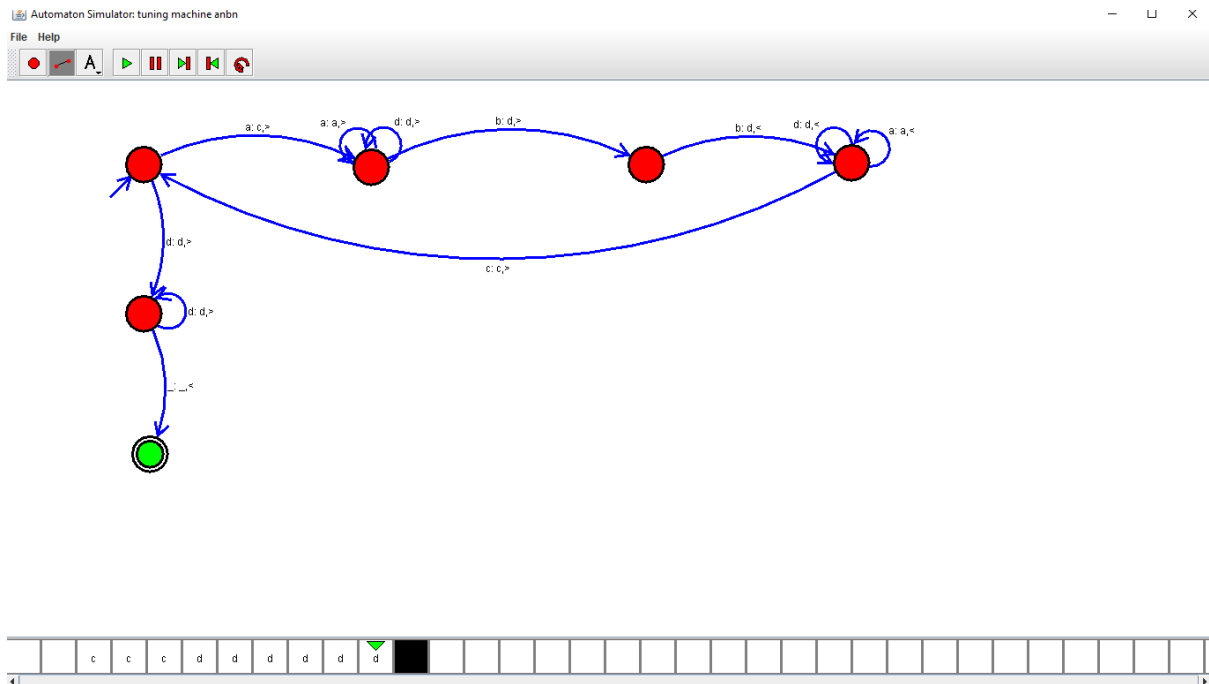
14.Design PDA using simulator to accept the input string $a^n b^{2n}$



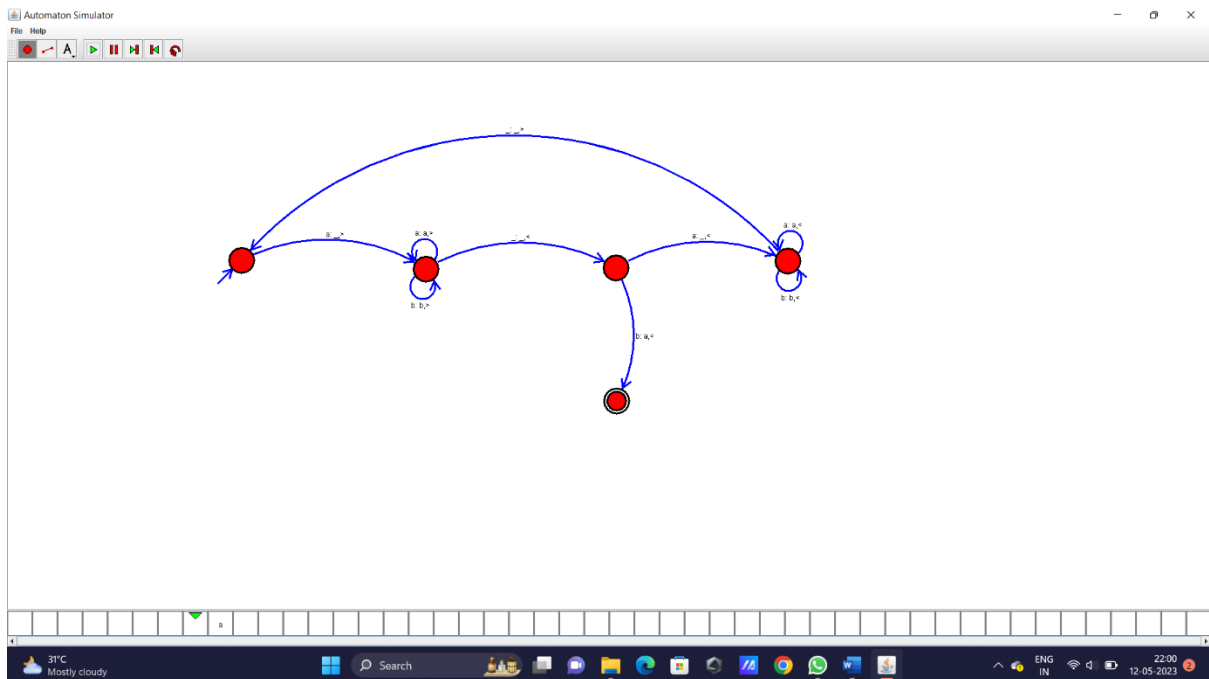
15.Design TM using simulator to accept the input string $a^n b^n$



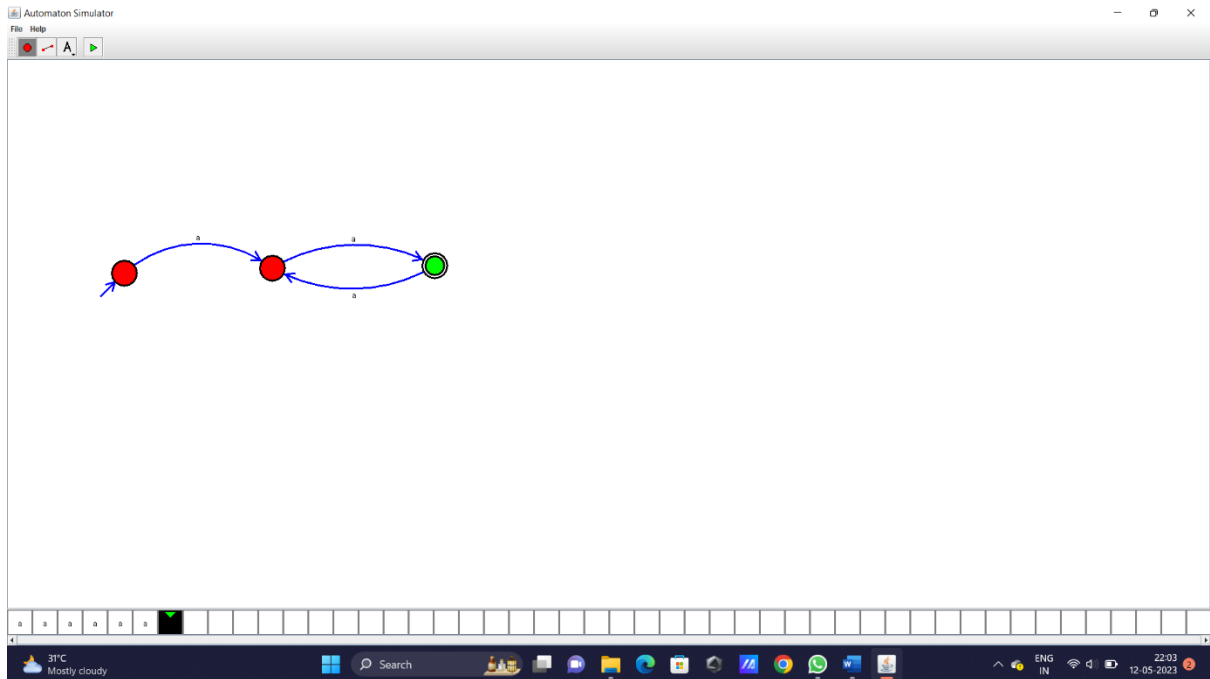
16.Design TM using simulator to accept the input string a^nb^{2n}



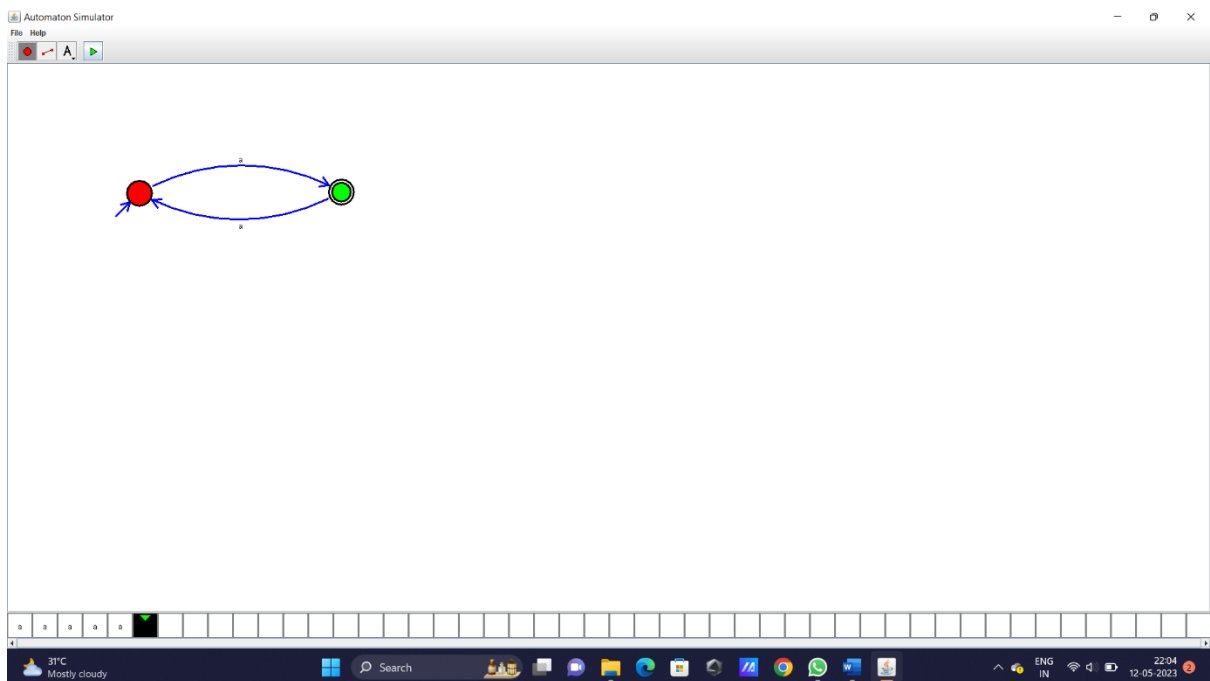
20.Design TM using simulator to perform subtraction of $aaa-aa$



21.Design DFA using simulator to accept even number of a's

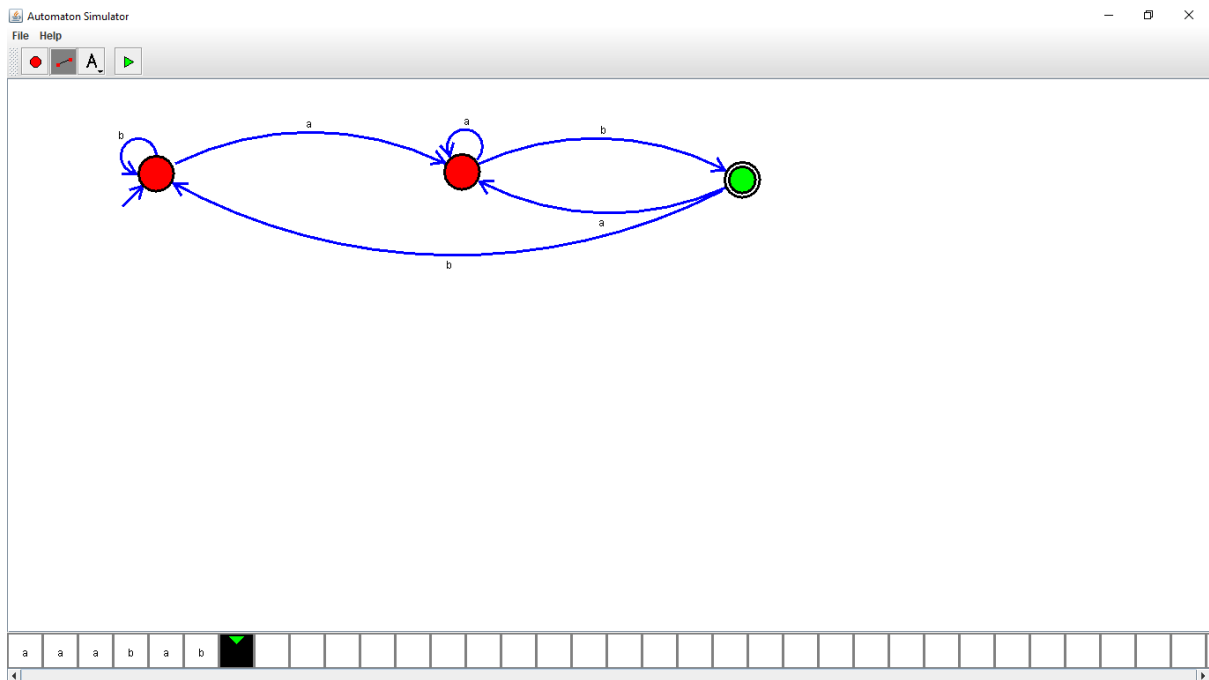


22.Design DFA using simulator to accept odd number of a's

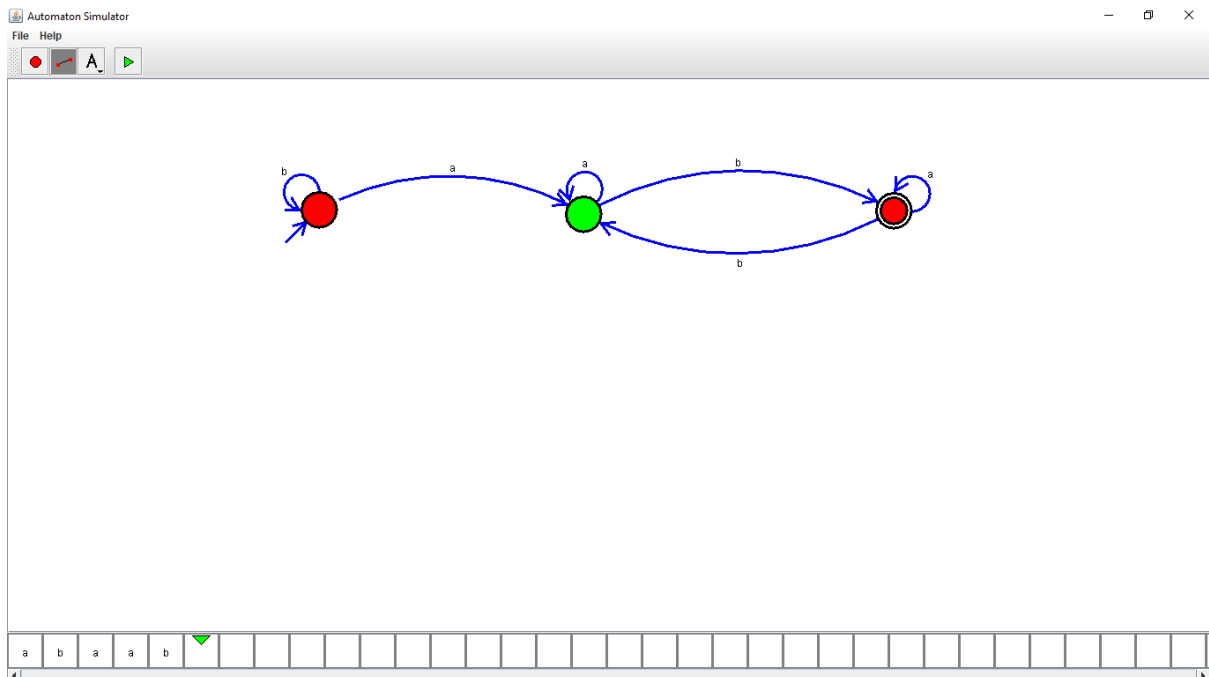


23.Design DFA using simulator to accept the string the end with ab over set {a,b}

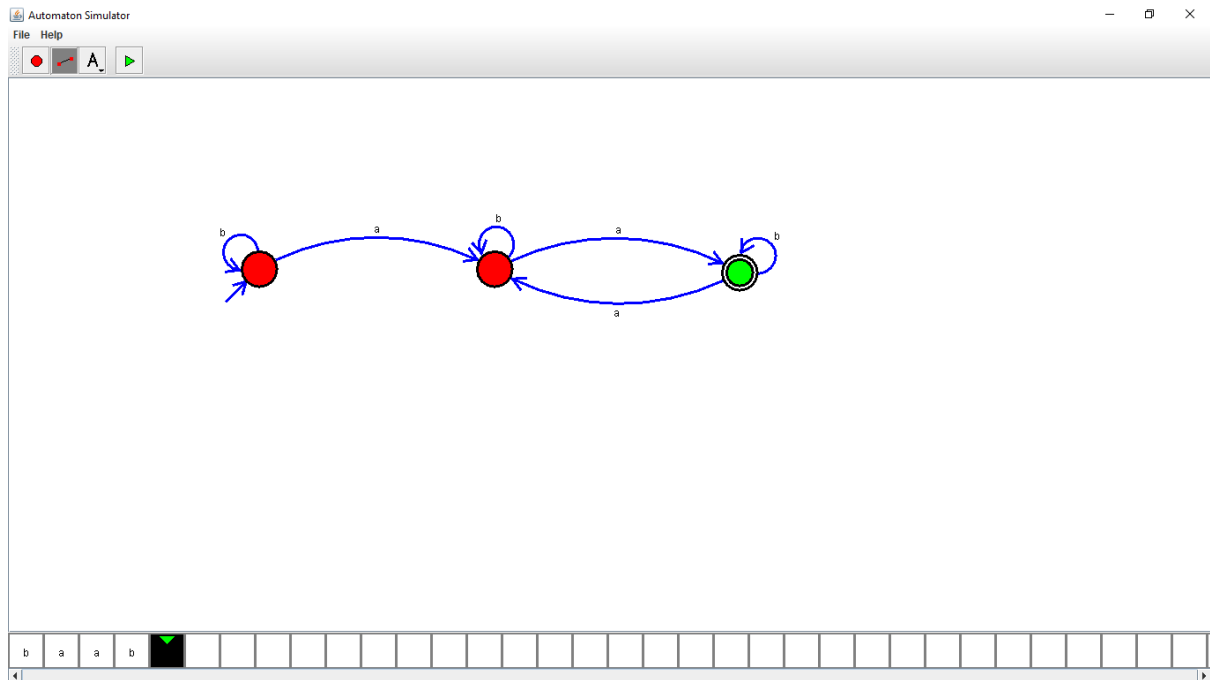
W= aaabab



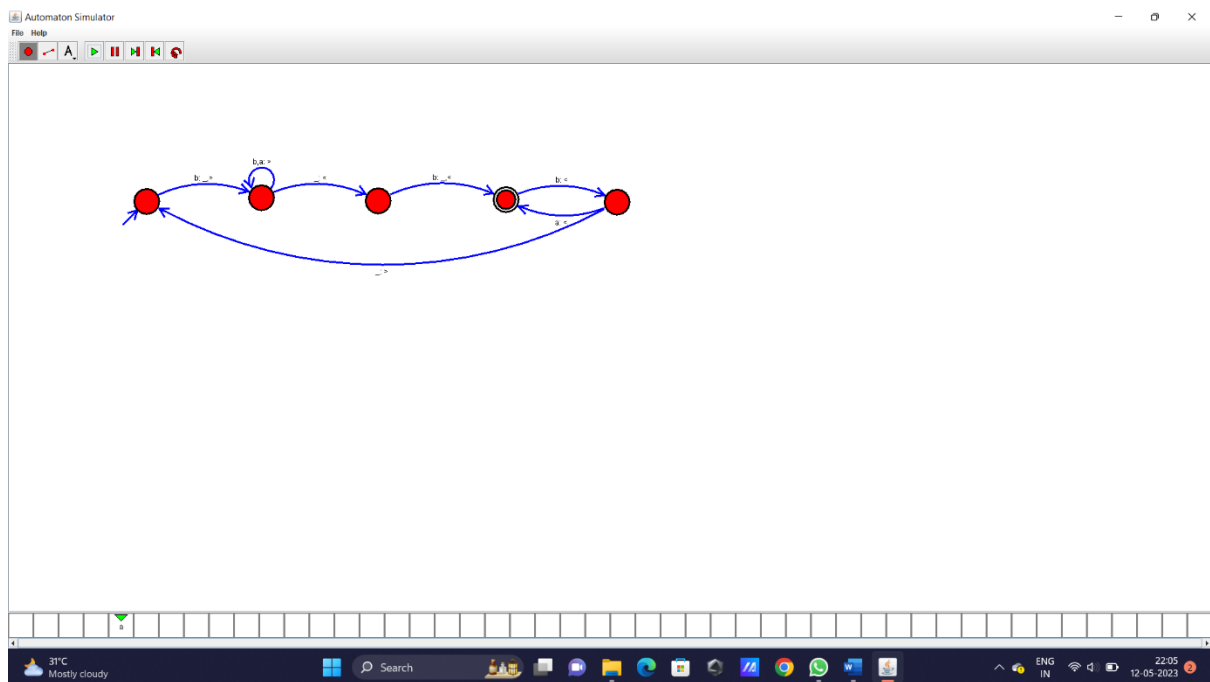
24.Design DFA using simulator to accept the string having 'ab' as substring over the set {a,b}



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

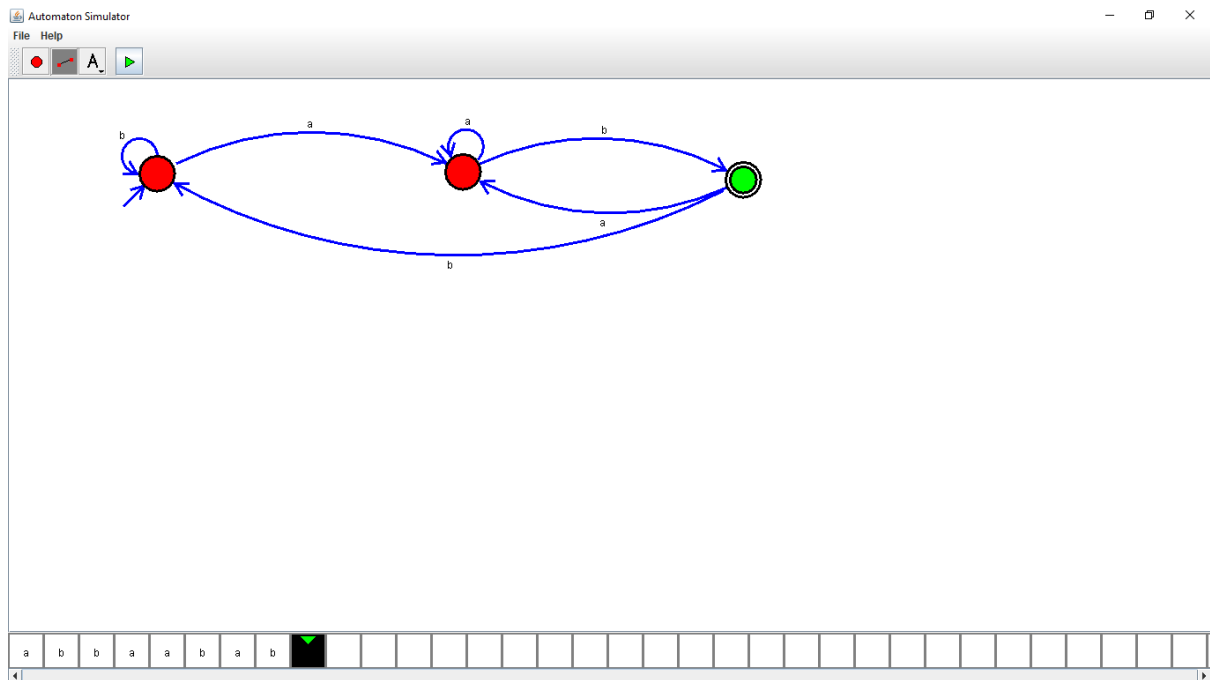


26.Design TM using simulator to accept the input string Palindrome bbabb

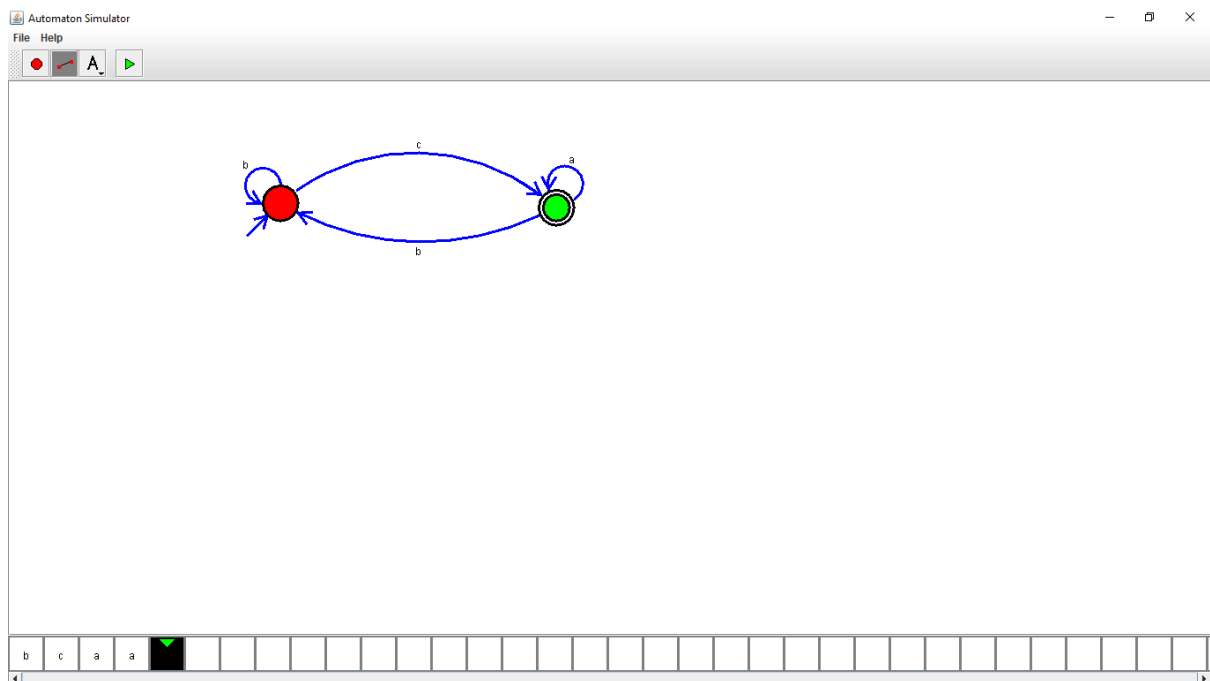


28.Design DFA using simulator to accept the string the end with ab over set {a,b}

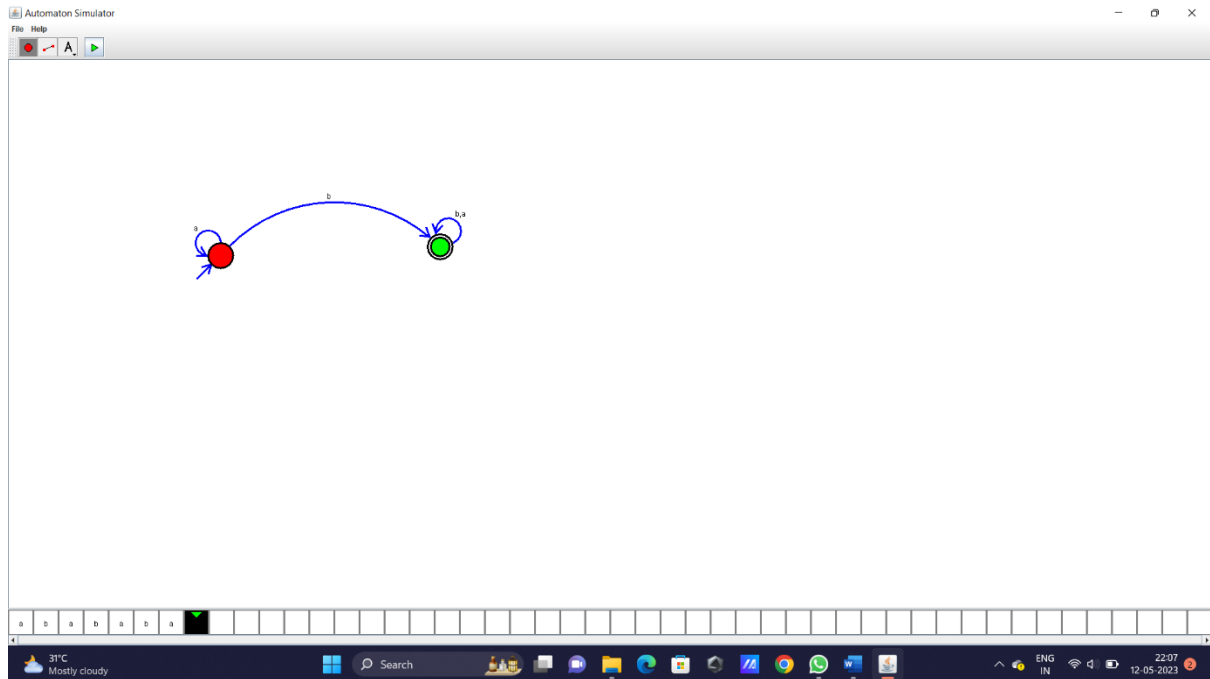
W= abbaabab



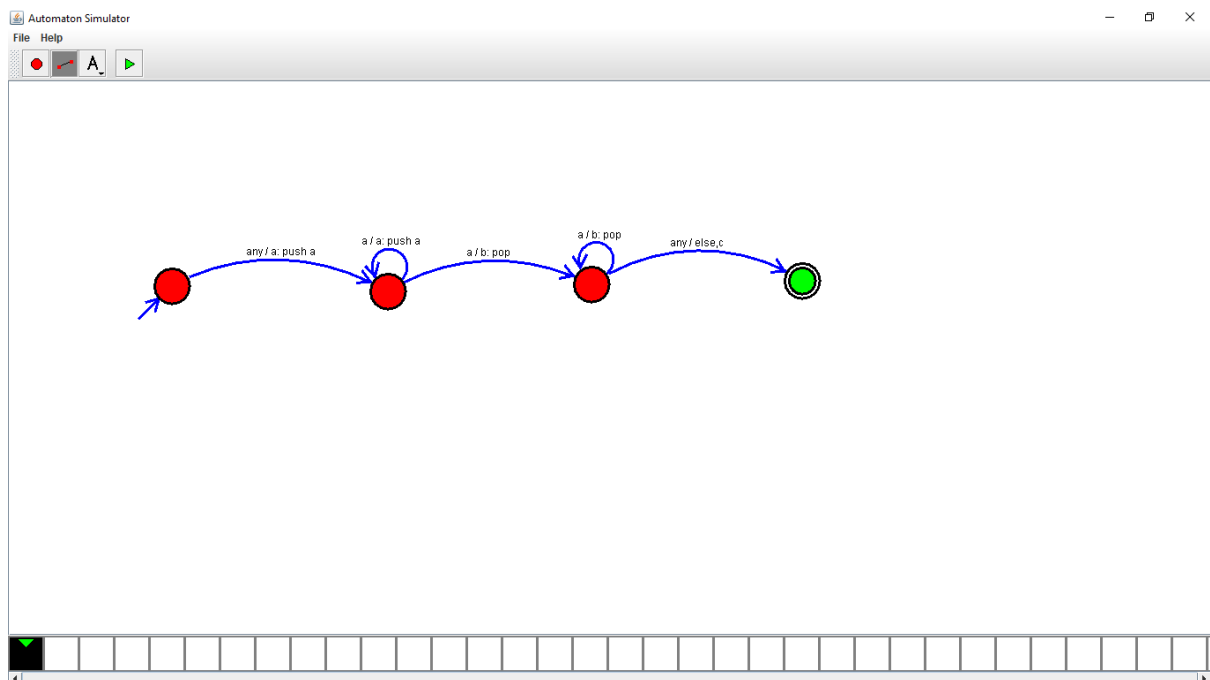
29.Design DFA using simulator to accept the input string "bc" ,"c",and "bcaaa".



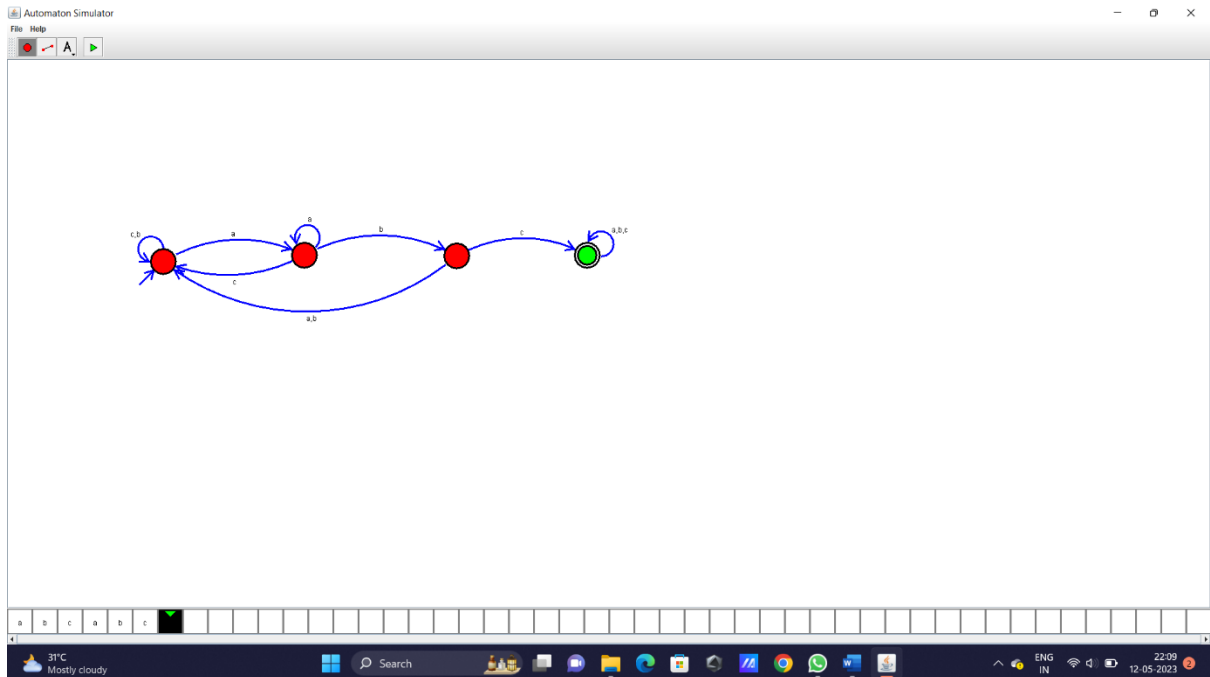
30.Design NFA to accept any number of a's where input={a,b}.



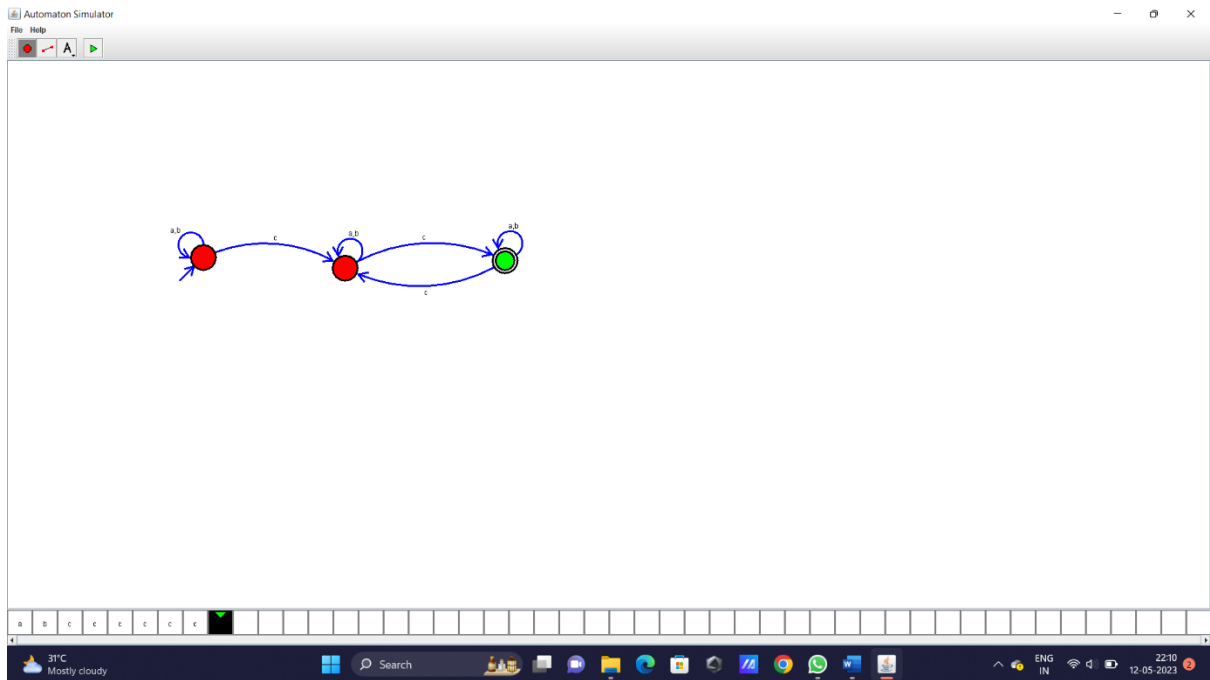
31.Design PDA using simulator to accept the input string $a^n b^n$



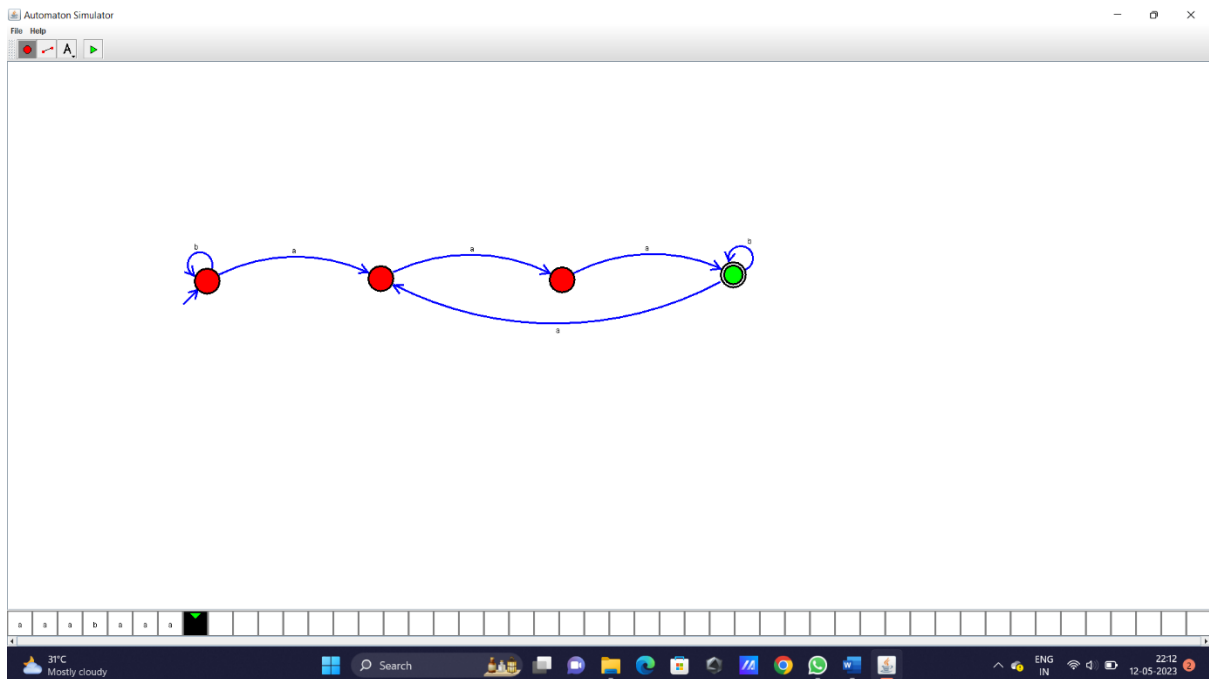
33.Design DFA using simulator to accept the string having 'abc' as substring over the set {a,b,c}



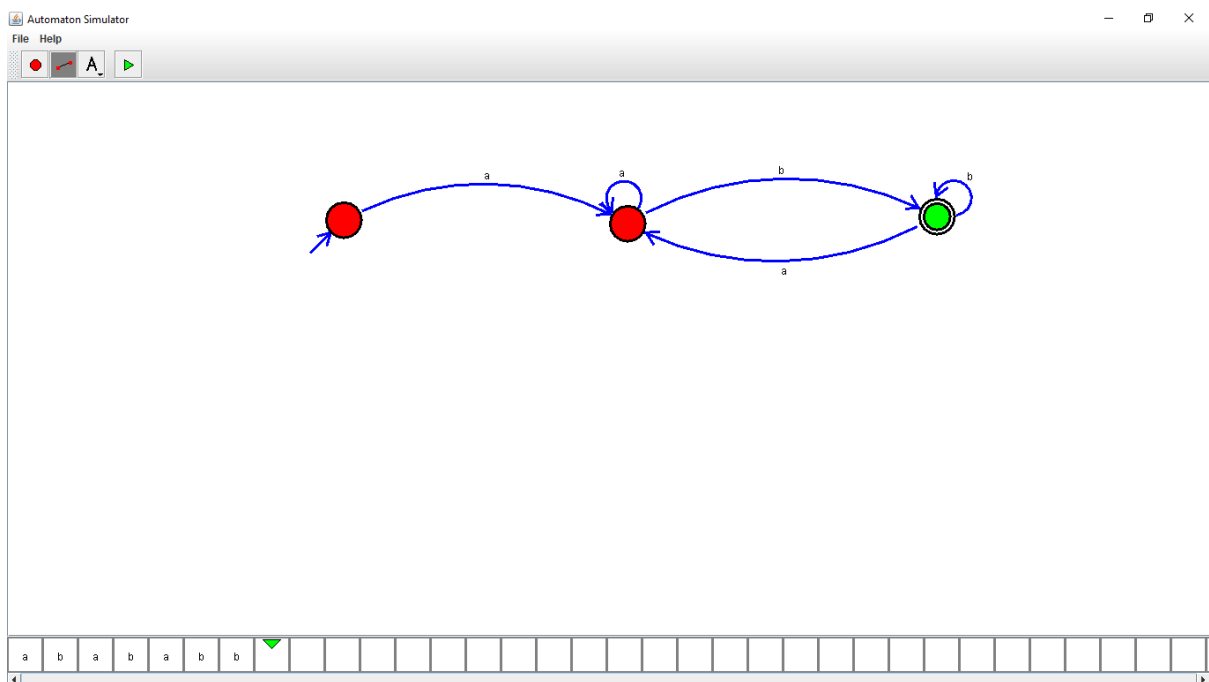
34.Design DFA using simulator to accept even number of c's over the set {a,b,c}



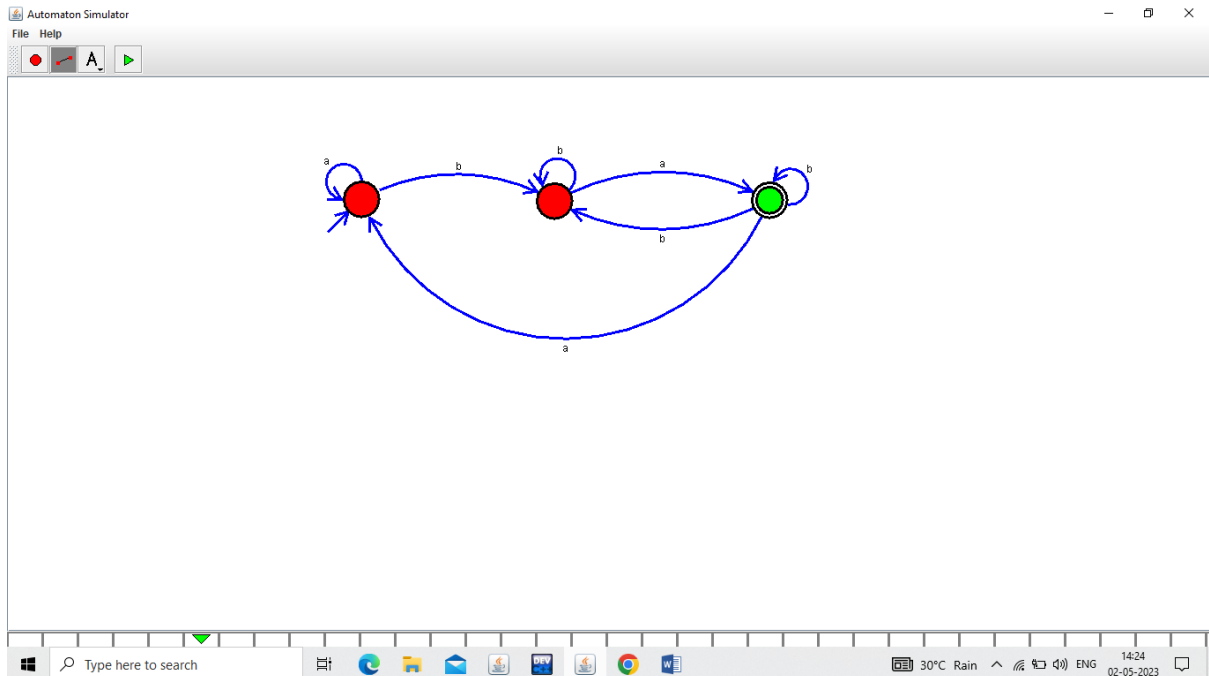
35.Design DFA using simulator to accept strings in which a's always appear tripled over input {a,b}



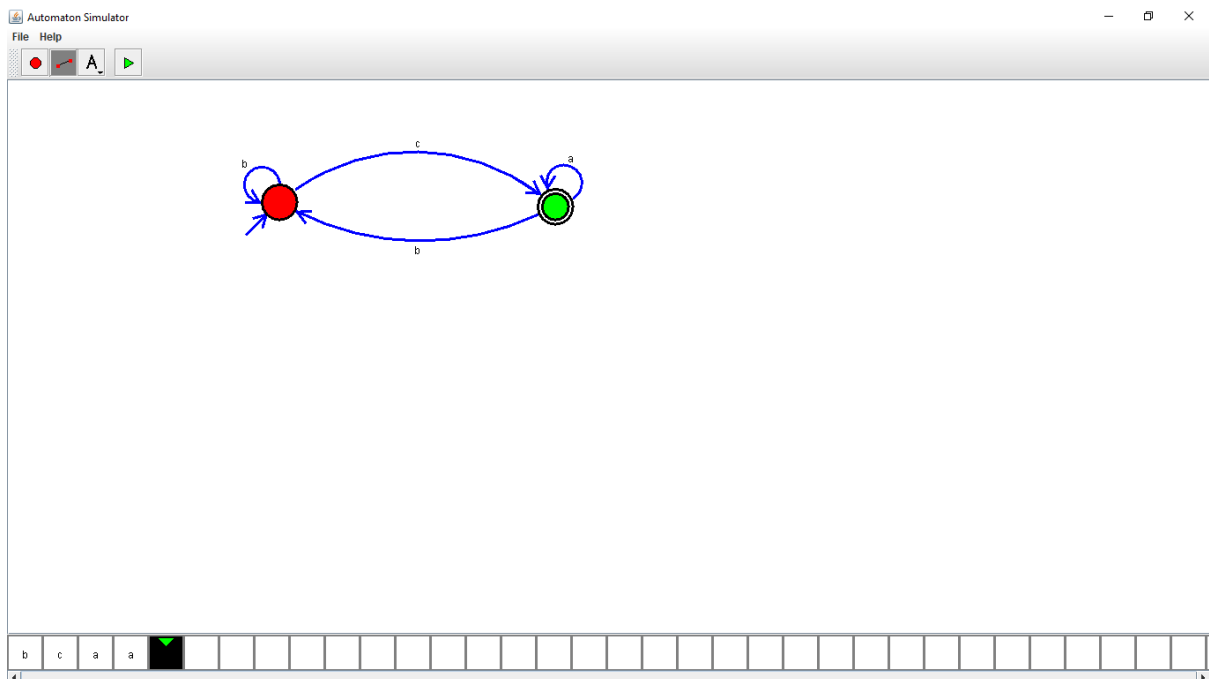
36.Design NFA using simulator to accept the string the start with a and end with b over set {a,b} and check W= abaab is accepted or not.



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

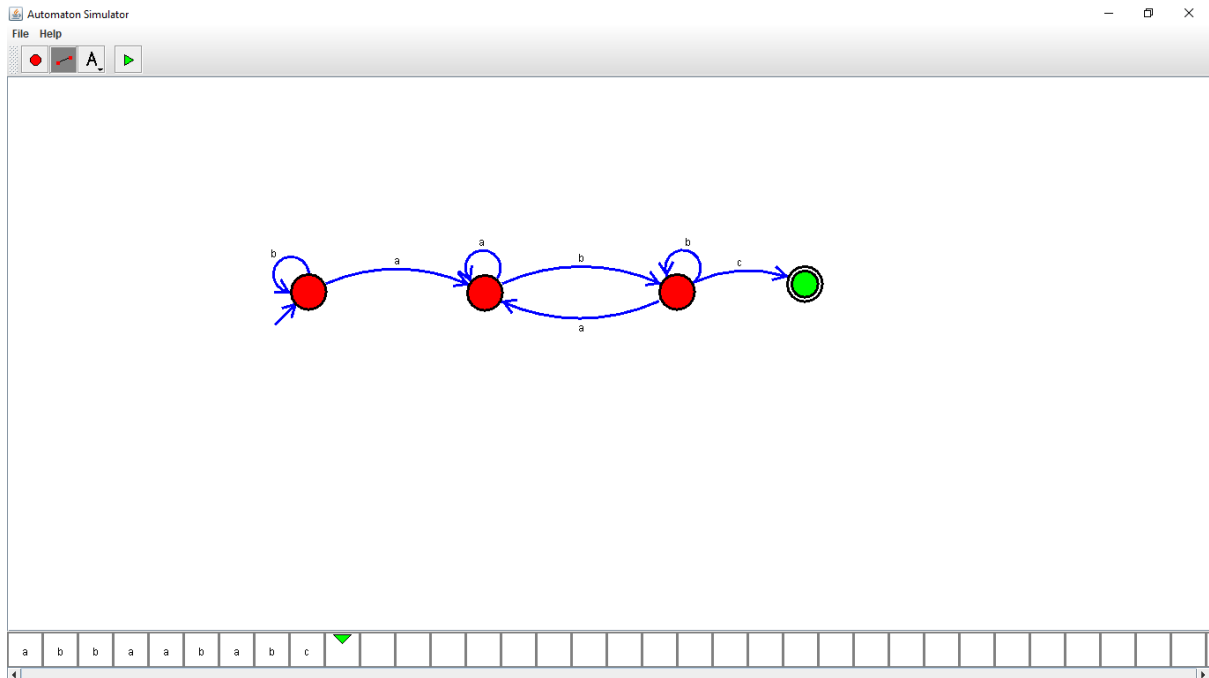


38.Design NFA using simulator to accept the input string "bbc" ,"c",and "bcaaa".



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

W= abbaababc



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

