



SAVEETHA
SCHOOL OF ENGINEERING

TOC EXPERIMENTS

COURSE CODE: CSA13

COURSE NAME: THEORY OF COMPUTATION

C PROGRAMMING:

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.

PROGRAM:

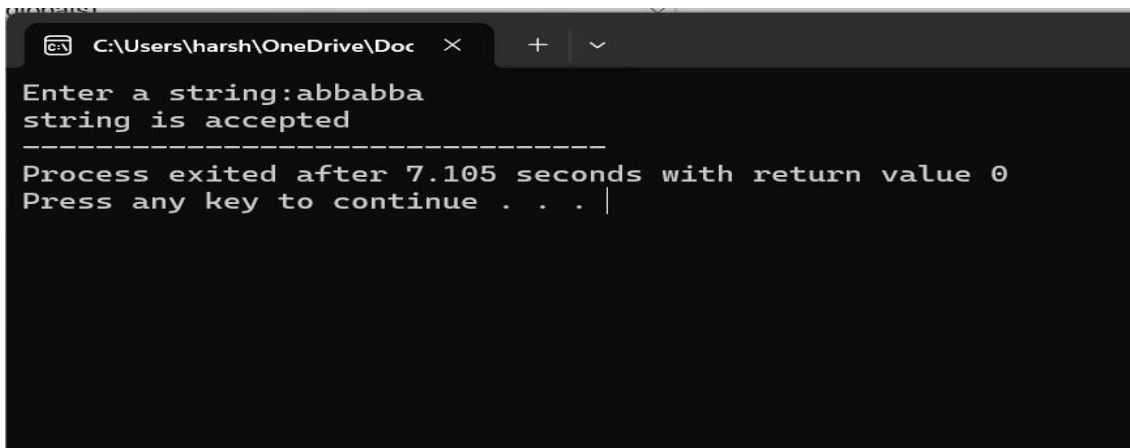
```
#include <stdio.h>
#include <string.h>
int main () {
    char String [100];
    printf ("Enter a string: ");
    scanf ("%s", String);
    if (String [0] == 'a' && String [strlen (String)-1] == 'a')
    {
        int i;
        for (i=0; i<strlen (String); i++) {
            if (String[i] == '0' || String[i] == '1')
            {
                printf ("Invalid! \n");
            }
        }
    }
}
```

```

        return 0;
    }
}
printf("string is accepted\n");
} else {
    printf("Not Accepted\n");
}
return 0;
}

```

OUTPUT:



```

C:\Users\harsh\OneDrive\Doc
Enter a string:abbabba
string is accepted
-----
Process exited after 7.105 seconds with return value 0
Press any key to continue . . . |

```

2. 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.

PROGRAM:

```

#include <stdio.h>
#include <string.h>
int main () {

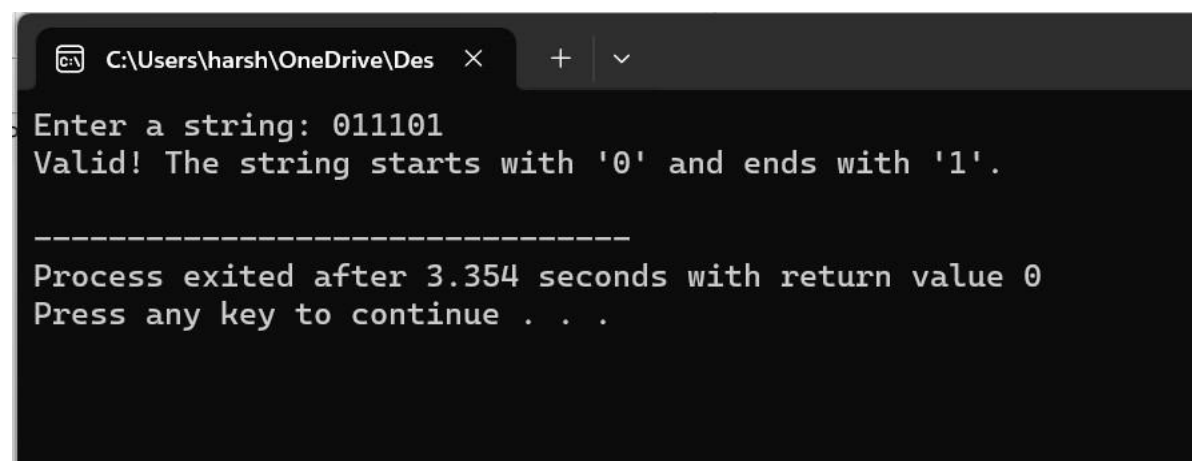
```

```

char String [100];
printf ("Enter a string: ");
scanf ("%s", String);
if (String [0] == '0' && String [strlen (String)-1] == '1')
{
    int i;
    for (i==; i<strlen (String); i++) {
        if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
                return 0;
            }
    }
    printf ("Valid! The string starts with '0' and ends with '1'. \n");
} else {
    printf ("Invalid! The string does not start with '0' and end with '1'. \n");
}
return 0;
}

```

OUTPUT:



```

C:\Users\harsh\OneDrive\Des > Enter a string: 011101
Valid! The string starts with '0' and ends with '1'.

-----
Process exited after 3.354 seconds with return value 0
Press any key to continue . . .

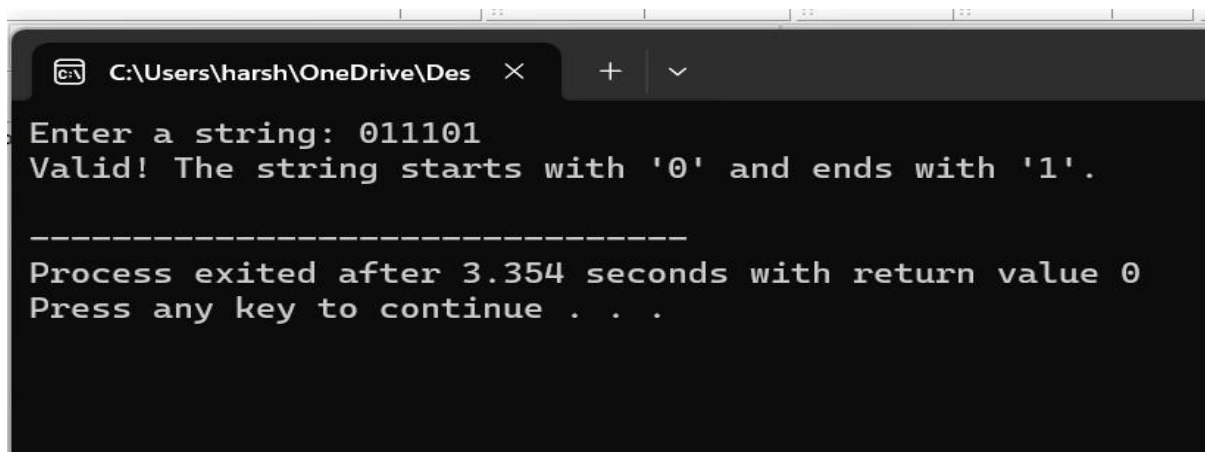
```

3. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) $S \rightarrow 0A1$ $A \rightarrow 0A \mid 1A \mid \epsilon$

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main () {
    char String [100];
    printf ("Enter a string: ");
    scanf ("%s", String);
    if (String [0] == '0' && String [strlen (String)-1] == '1')
    {
        int i;
        for (i==0; i<strlen (String); i++) {
            if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
                return 0;
            }
        }
        printf ("Valid! The string starts with '0' and ends with '1'.\n");
    } else {
        printf ("Invalid! The string does not start with '0' and end with '1'.\n");
    }
    return 0;
}
```

OUTPUT:



```
C:\Users\harsh\OneDrive\Des >
Enter a string: 011101
Valid! The string starts with '0' and ends with '1'.

-----
Process exited after 3.354 seconds with return value 0
Press any key to continue . . .
```

4. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \epsilon$.

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main () {
    char String [100];
    printf ("Enter a string: ");
    scanf ("%s", String);
    if (String [0] == '0' && String [strlen (String)-1] == '1' || String [0]
    == '1' && String [strlen (String)-1] == '1'))
    {
        int i;
        for (i==; i<strlen (String); i++) {
            if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
```

```

        return 0;
    }
}

printf("Valid! The string starts with '0' and ends with '0' or starts with '1'
and ends with '1'.\n");

} else {

    printf("Invalid! The string starts with '0' and ends with '0' or starts with '1'
and ends with '1'.\n");

}

return 0;
}

```

OUTPUT:

```

C:\Users\harsh\OneDrive\Des X + v
Enter a string: 010010
Valid! The string starts with '0' and ends with '0' or starts
-----
Process exited after 3.735 seconds with return value 0
Press any key to continue . . .

C:\Users\harsh\OneDrive\Des X + v
Enter a string: 101001
Valid! The string starts with '0' and ends with '0' or starts with '1' and
ends with '1'.
-----
Process exited after 2.97 seconds with return value 0
Press any key to continue . . .

```

5. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) $S \rightarrow 0S0 \mid A$

$A \rightarrow 1A \mid \epsilon$

PROGRAM:

```
#include <stdio.h>

#include <string.h>

int main () {

    char String [100];

    printf ("Enter a string: ");

    scanf ("%s", String);

    if (String [0] == '0' && String [strlen (String)-1] == '1' || String [0]
== '1' && String [strlen (String)-1] == '1')

        {

            int i;

            for (i==; i<strlen (String); i++) {

                if (String[i]<'0' || String[i]>'1')

                    {

                        printf ("Invalid! \n");

                        return 0;

                    }

            }

            printf ("Valid! The string starts with '0' and ends with '0' or starts with '1'
and ends with '1'. \n");

        } else {

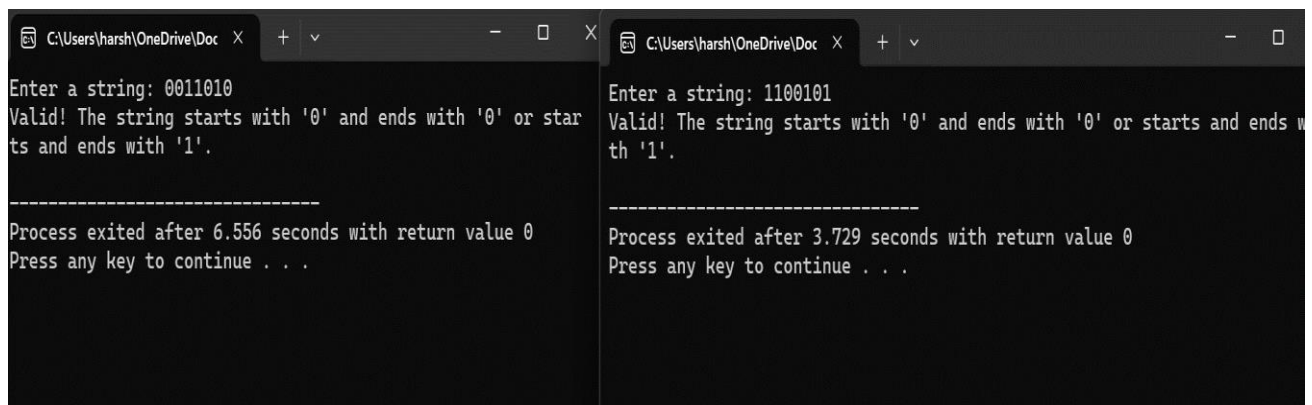
            printf ("Invalid! The string starts with '0' and ends with '0' or starts with '1'
and ends with '1'. \n");

        }

        return 0;

    }
```

OUTPUT:



The image shows two side-by-side terminal windows. The left window shows the program being executed with the input string "0011010". The output is "Valid! The string starts with '0' and ends with '0' or starts and ends with '1'." followed by a separator line and "Process exited after 6.556 seconds with return value 0". The right window shows the program being executed with the input string "1100101". The output is "Valid! The string starts with '0' and ends with '0' or starts and ends with '1'." followed by a separator line and "Process exited after 3.729 seconds with return value 0".

```
C:\Users\harsh\OneDrive\Doc x + - □ X
Enter a string: 0011010
Valid! The string starts with '0' and ends with '0' or starts and ends with '1'.

-----
Process exited after 6.556 seconds with return value 0
Press any key to continue . . .

C:\Users\harsh\OneDrive\Doc x + - □ X
Enter a string: 1100101
Valid! The string starts with '0' and ends with '0' or starts and ends with '1'.

-----
Process exited after 3.729 seconds with return value 0
Press any key to continue . . .
```

6. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) $S \rightarrow 0S1 \mid \epsilon$

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main () {
    char String [100];
    printf ("Enter a string: ");
    scanf ("%s", String);
    if (String [0] == '0' && String [strlen (String)-1] == '1')
    {
        int i;
        for (i==; i<strlen (String); i++) {
            if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
                return 0;
            }
        }
    }
}
```

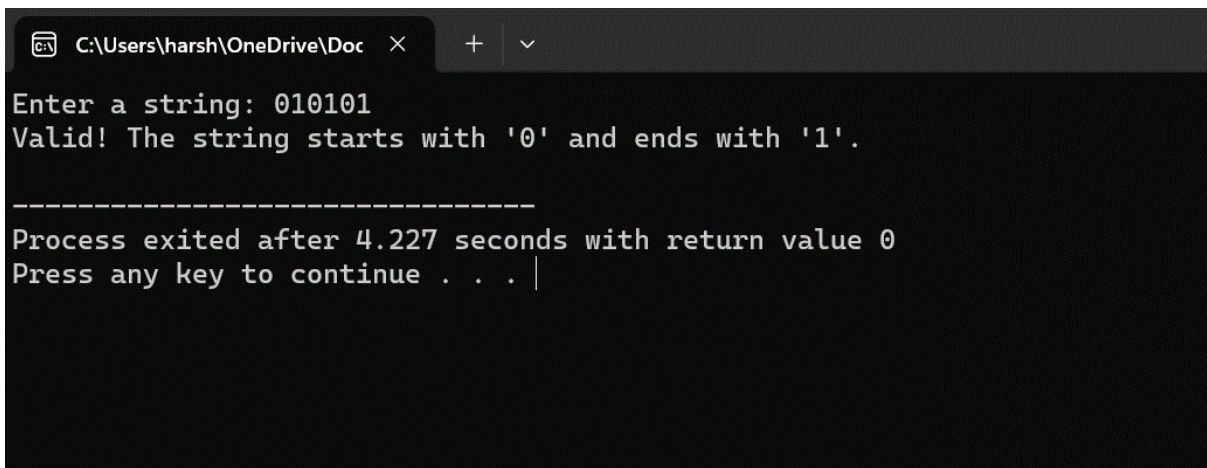


```

    }
}
printf("Valid! The string starts with '0' and ends with '1'.\n");
} else {
    printf("Invalid! The string does not start with '0' and end with '1'.\n");
}
return 0;

```

OUTPUT:



```

C:\Users\harsh\OneDrive\Doc >
Enter a string: 010101
Valid! The string starts with '0' and ends with '1'.

-----
Process exited after 4.227 seconds with return value 0
Press any key to continue . . . |

```

7. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) $S \rightarrow A101A$, $A \rightarrow 0A \mid 1A \mid \epsilon$

PROGRAM:

```

#include <stdio.h>

#include <string.h>

int main () {

    char String [100];

    printf ("Enter a string: ");

```

```

scanf ("%s", String);
if (String [0] == '0' || '1' &&String [strlen (String)-1] == '0' || '1')
{
    int i;
    for (i==; i<strlen (String); i++) {
        if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
                return 0;
            }
    }
    printf ("Valid! The string starts with '0' or '1' and ends with '0' or '1'. \n");
} else {
    printf ("Invalid! The string does not start with '0' or '1' and end with '0' or
'1'. \n");
}
return 0;
}

```

OUTPUT:

The image shows two side-by-side screenshots of a C++ IDE window. The window title is 'C:\Users\harsh\OneDrive\Doc'. The left screenshot shows the input 'Enter a string: 10010010' and the output 'Valid! The string starts with '0' or '1' and ends with '0' or '1'.' The right screenshot shows the input 'Enter a string: 010101001' and the output 'Valid! The string starts with '0' or '1' and ends with '0' or '1'.' Both screenshots show the process exiting after a few seconds with a return value of 0.

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>

int main () {

    char String [100];

    printf ("Enter a string: ");

    scanf ("%s", String);

    if (String [0] =='a'&&String [strlen (String)-1] =='a')

        {

            int i;

            for (i=0; i<strlen (String); i++) {

                if (String[i]=='0'||String[i]=='1')

                    {

                        printf ("Invalid! \n");

                        return 0;

                    }

            }

            printf ("Accepted\n");

        } else {

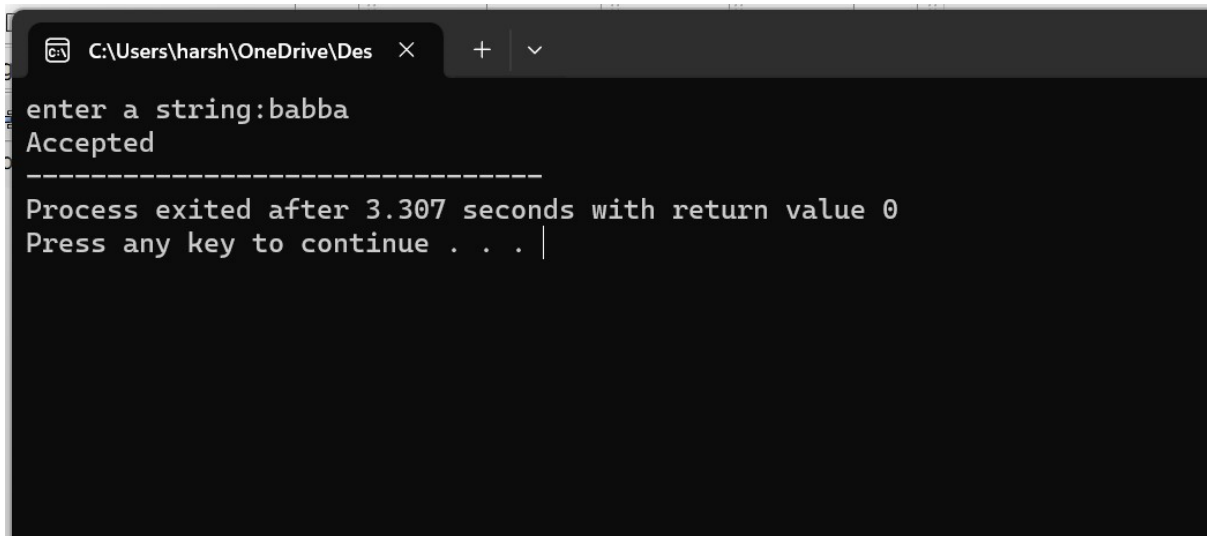
            printf ("Not Accepted\n");

        }

    return 0;

}
```

OUTPUT:



```
C:\Users\harsh\OneDrive\Des
enter a string: babba
Accepted
-----
Process exited after 3.307 seconds with return value 0
Press any key to continue . . . |
```

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

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main () {
    char String [100];
    printf ("Enter a string: ");
    scanf ("%s", String);
    if (String [0] == '0' && String [strlen (String)-1] == '1')
    {
        int i;
        for (i==; i<strlen (String); i++) {
            if (String[i]<'0' || String[i]>'1')
            {
                printf ("Invalid! \n");
            }
        }
    }
}
```

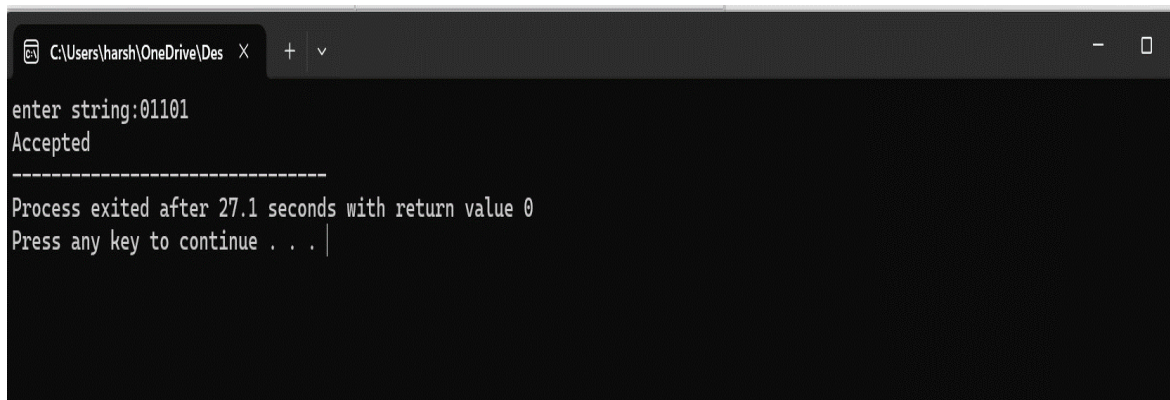
```

        return 0;
    }
}

printf("Valid! The string starts with '0' and ends with '1'.\n");
} else {
    printf("Invalid! The string does not start with '0' and end with '1'.\n");
}
return 0;
}

```

OUTPUT:



```

C:\Users\harsh\OneDrive\Desktop X + v
enter string:01101
Accepted
-----
Process exited after 27.1 seconds with return value 0
Press any key to continue . . .

```

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>

int main () {
    int n;

```

```

int m;
printf ("Enter the number of states: ");
scanf ("%d", &n);
printf ("Enter the number of transitions: ");
scanf ("%d", &m);
int transitions [3][3];
for (int i = 0; i < m; i++) {
    printf ("Enter transition %d (fromState inputSymbol toState): ", i
+ 1);
    scanf ("%d", &transitions[i][0]);
    char inputSymbol [2];
    scanf ("%1s", inputSymbol);
    scanf ("%d", &transitions[i][2]);
    if (inputSymbol [0] == 'e') {
        transitions[i][1] = 'e';
    } else {
        transitions[i][1] = inputSymbol [0];
    }
}
for (int i = 0; i < n; i++) {
    printf("e-closure(%d): {%d ", i, i);
    for (int j = 0; j < m; j++) {
        if (transitions[j][0] == i && transitions[j][1] == 'e') {
            printf ("%d ", transitions[j][2]);
        }
    }
}

```

```

    }

    printf("{ \n");

}

return 0;

}

```

OUTPUT:

```

C:\Users\harsh\OneDrive\Des
Enter the number of states: 4
Enter the number of transitions: 3
Enter transition 1 (fromState inputSymbol toState): 0
e
1
Enter transition 2 (fromState inputSymbol toState): 1
a
3
Enter transition 3 (fromState inputSymbol toState): 2
e
3
e-closure(0): { 0 1 }
e-closure(1): { 1 }
e-closure(2): { 2 3 }
e-closure(3): { 3 }

-----
Process exited after 25.6 seconds with return value 0
Press any key to continue . . .

```

11. 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>

int main () {

    int n;

    int m;

    printf ("Enter the number of states: ");

    scanf ("%d", &n);

```

```

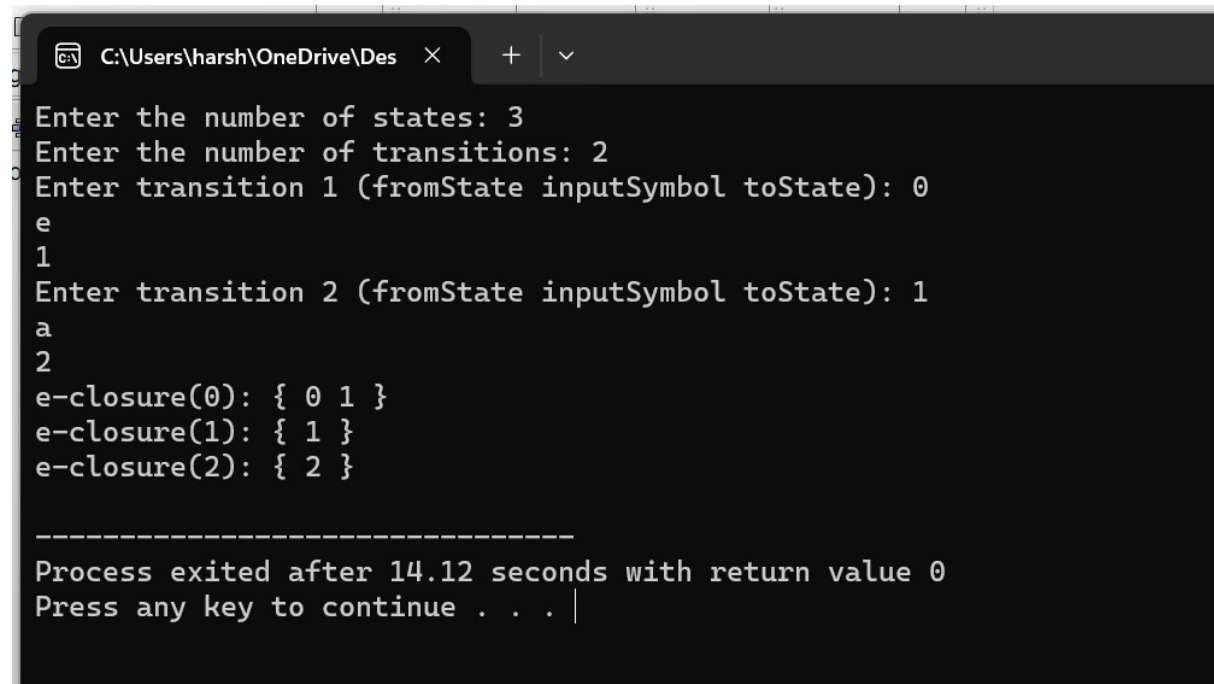
printf ("Enter the number of transitions: ");
scanf ("%d", &m);
int transitions [3][3];
for (int i = 0; i < m; i++) {
    printf ("Enter transition %d (fromState inputSymbol toState): ", i
+ 1);
    scanf ("%d", &transitions[i][0]);
    char inputSymbol [2];
    scanf ("%1s", inputSymbol);
    scanf ("%d", &transitions[i][2]);
    if (inputSymbol [0] == 'e') {
        transitions[i][1] = 'e';
    } else {
        transitions[i][1] = inputSymbol [0];
    }
}
for (int i = 0; i < n; i++) {
    printf("e-closure(%d): { %d ", i, i);
    for (int j = 0; j < m; j++) {
        if (transitions[j][0] == i && transitions[j][1] == 'e') {
            printf ("%d ", transitions[j][2]);
        }
    }
    printf (“} \n”);
}

```



```
    return 0;
}
```

OUTPUT:

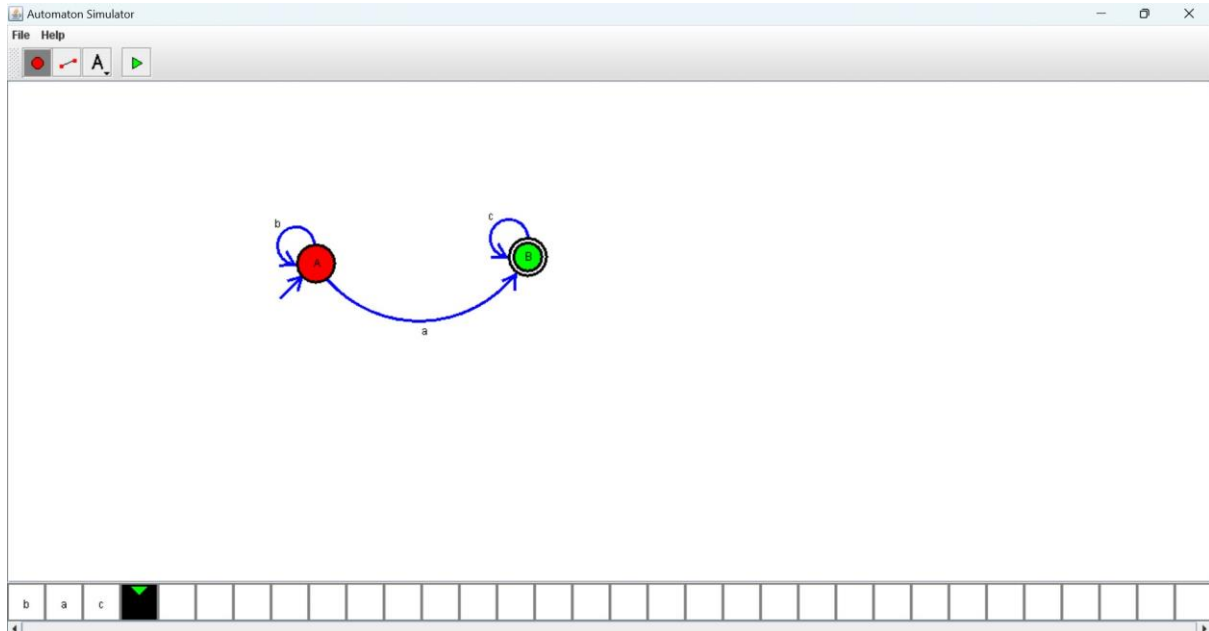


```
C:\Users\harsh\OneDrive\Des >
Enter the number of states: 3
Enter the number of transitions: 2
Enter transition 1 (fromState inputSymbol toState): 0
e
1
Enter transition 2 (fromState inputSymbol toState): 1
a
2
e-closure(0): { 0 1 }
e-closure(1): { 1 }
e-closure(2): { 2 }

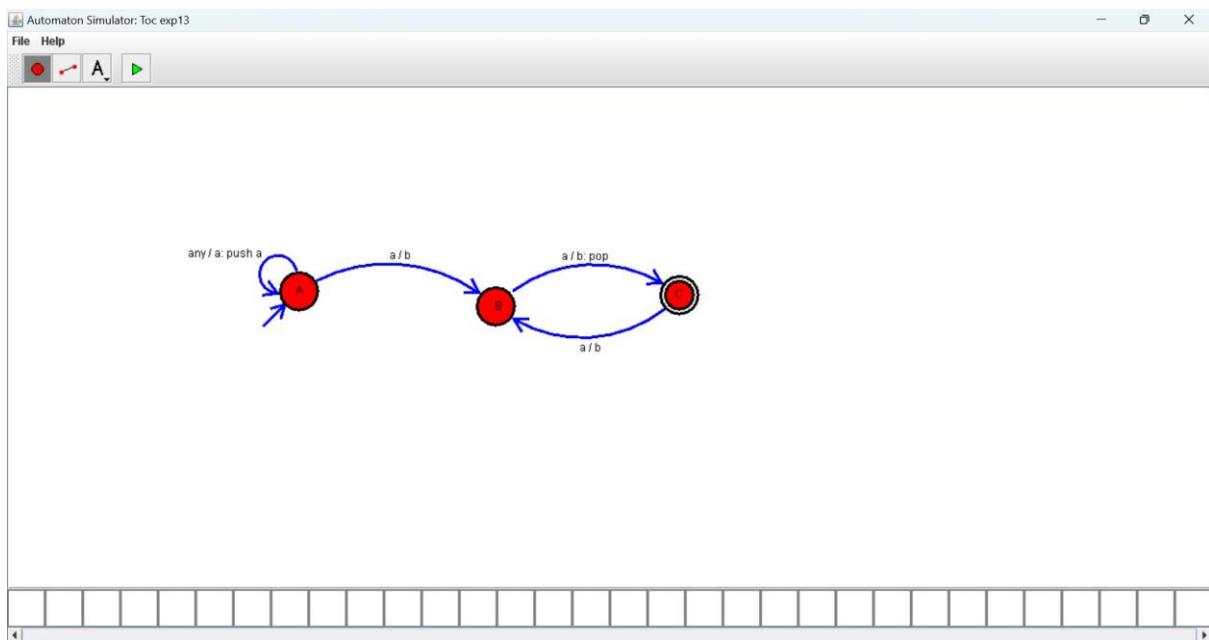
-----
Process exited after 14.12 seconds with return value 0
Press any key to continue . . . |
```

AUTO SIM:

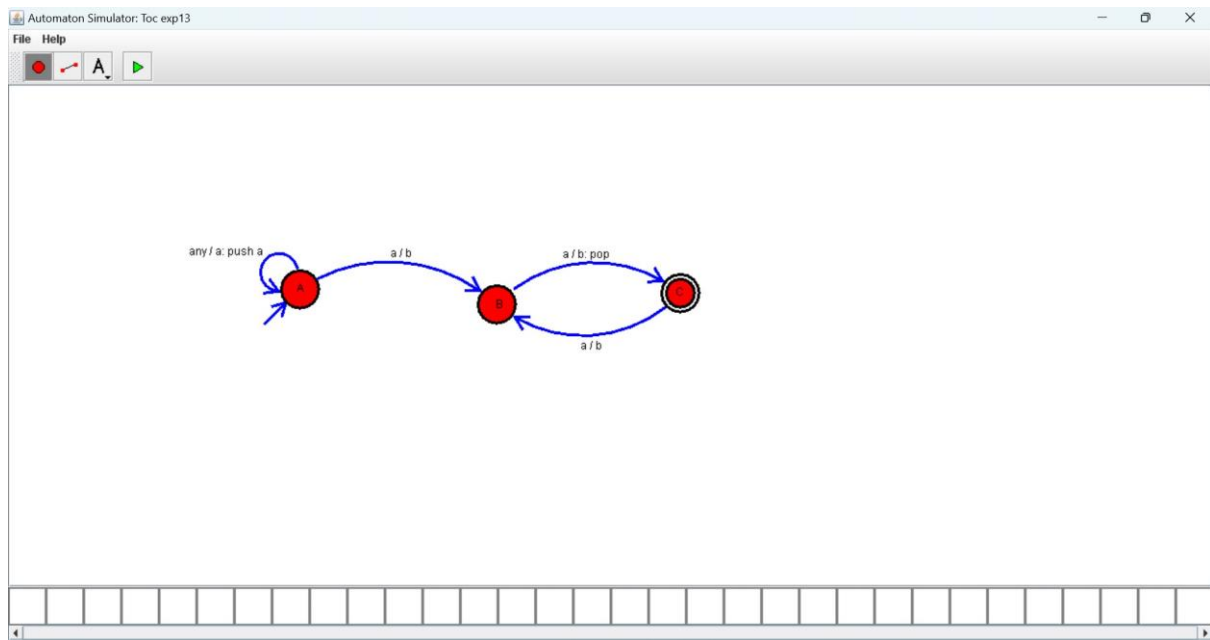
12. Design DFA using simulator to accept the input string “a”, “a” and” bac”



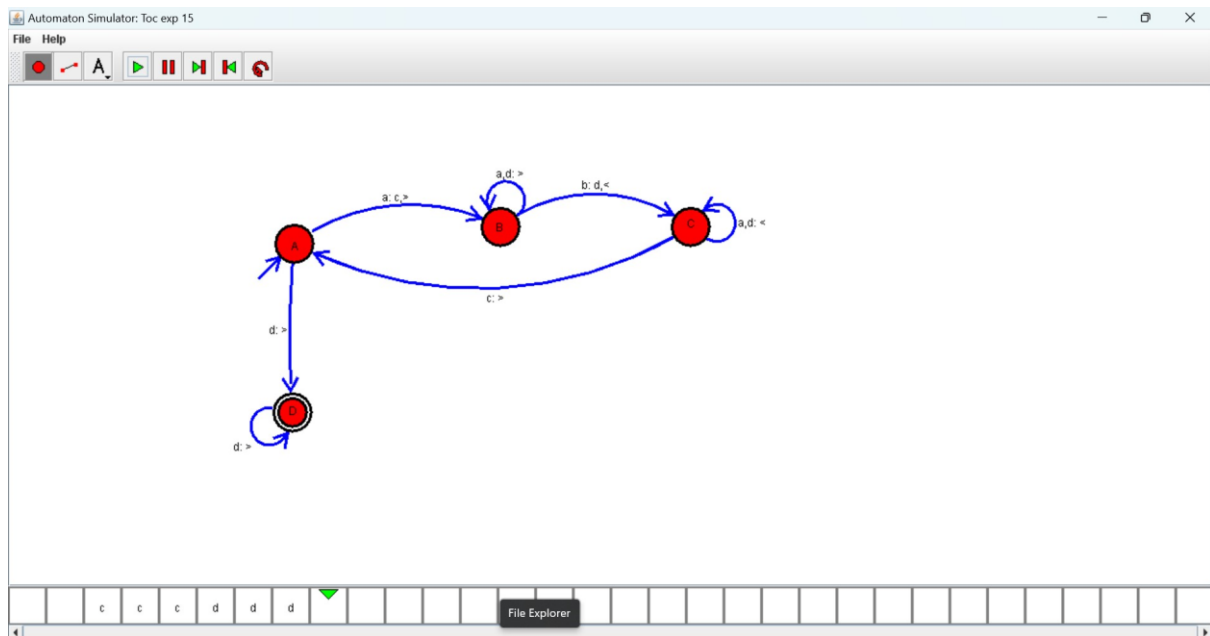
13. Design PDA using simulator to accept the input string aabb



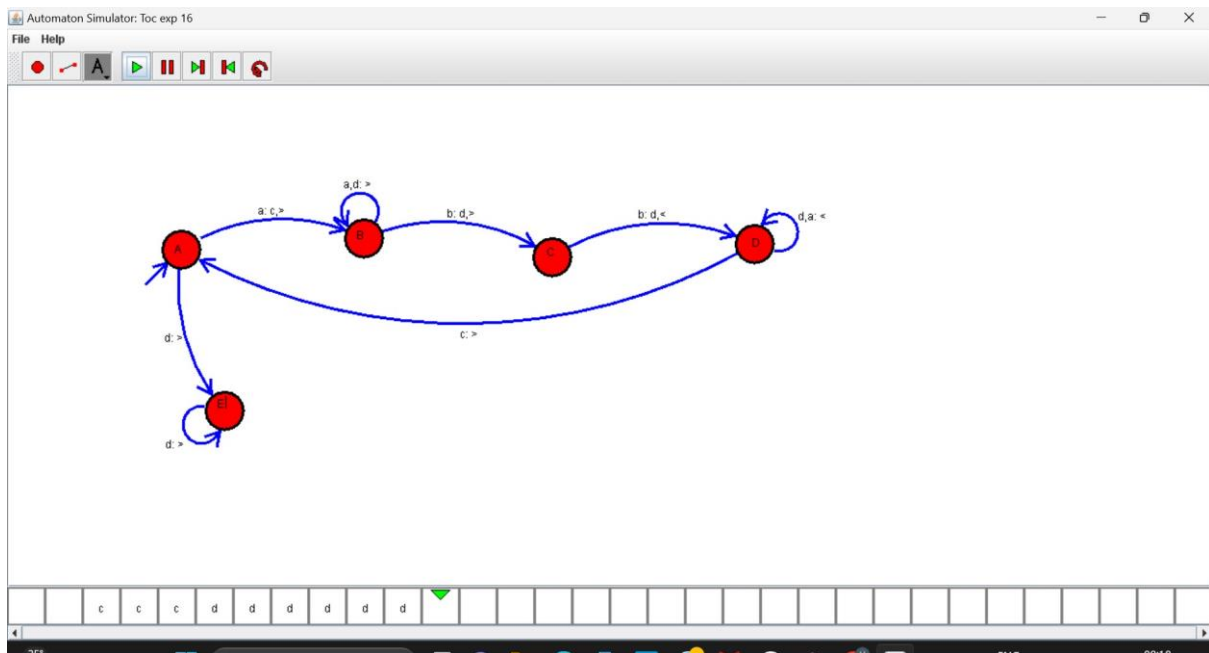
14. Design PDA using simulator to accept the input string $a^n b^{2n}$



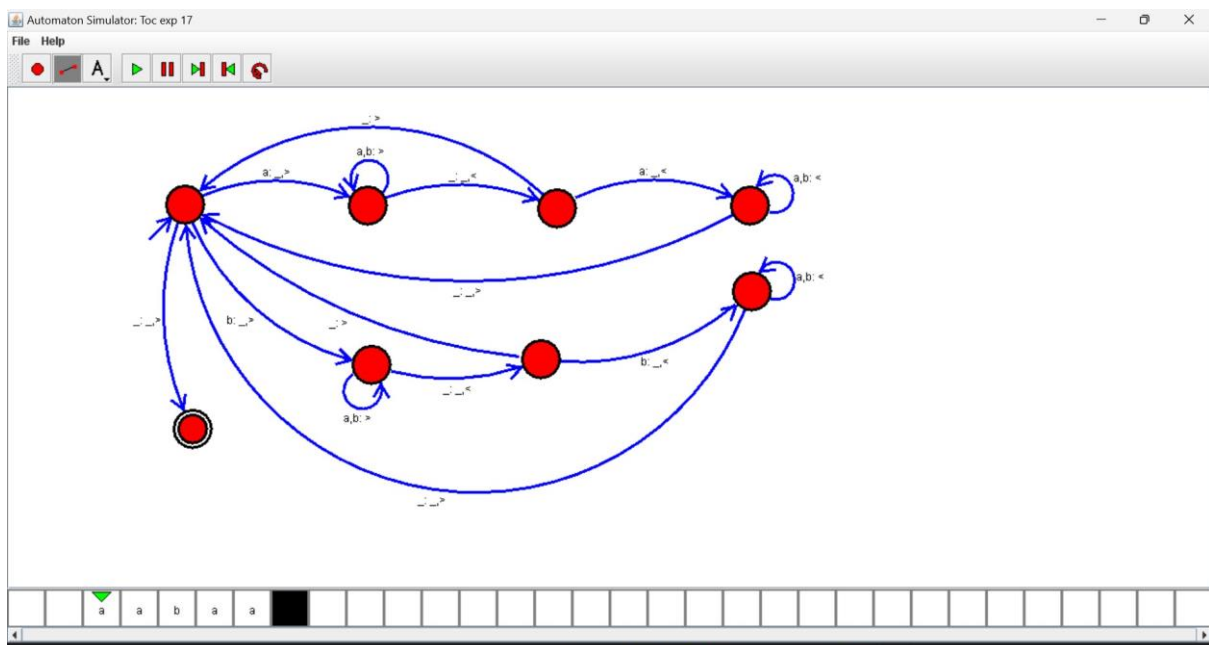
15. Design TM using simulator to accept the input string $a^n b^n$



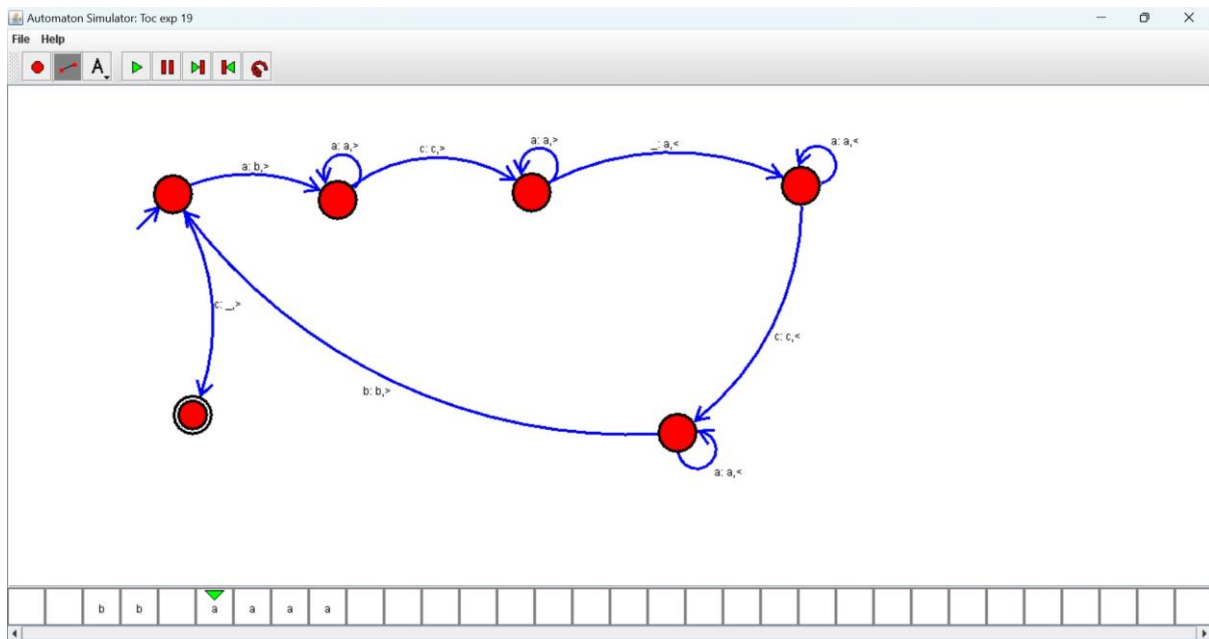
16. Design TM using simulator to accept the input string a^nb^{2n}



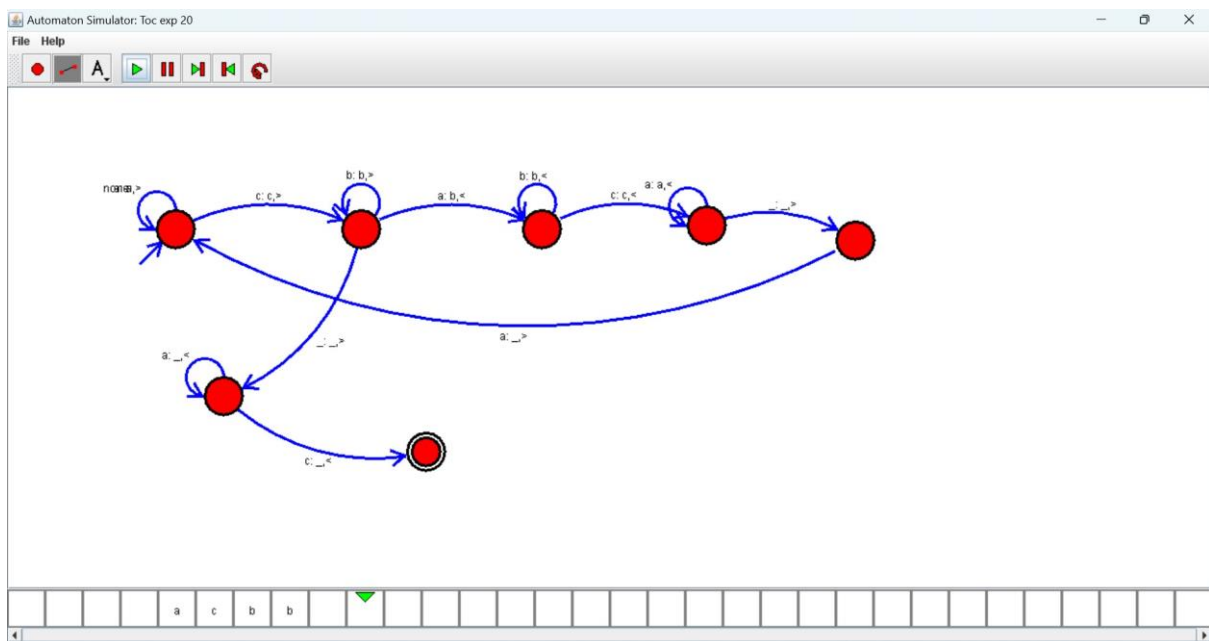
17. Design TM using simulator to accept the input string Palindrome ababa



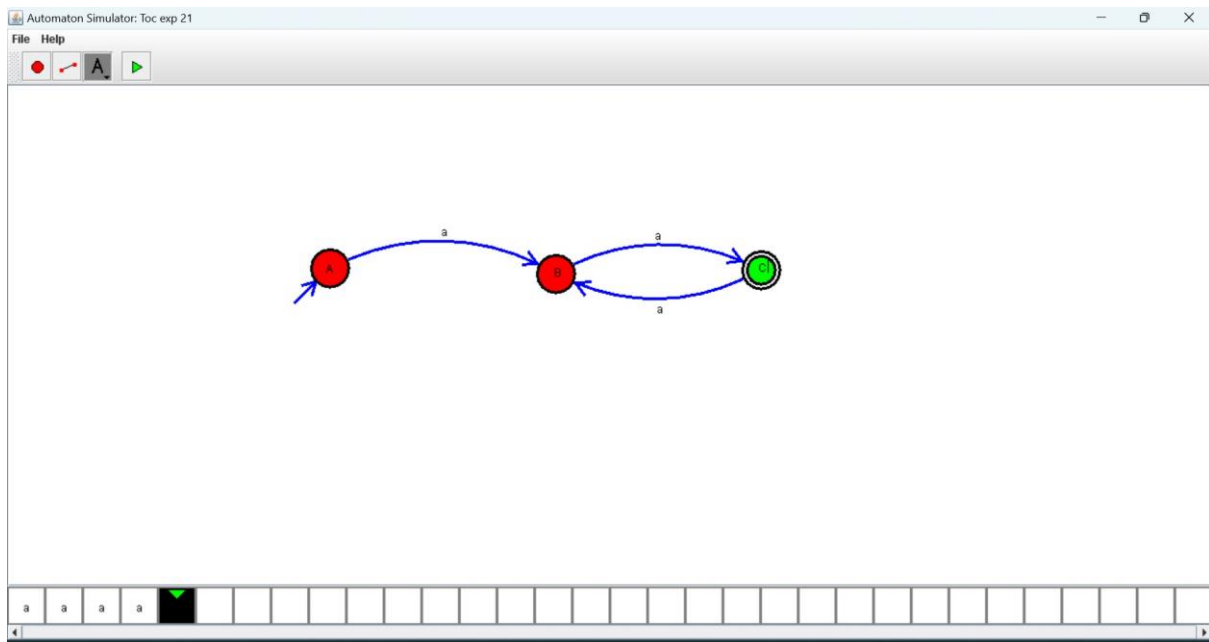
19. Design TM using simulator to perform addition of 'aa' and 'aaa'



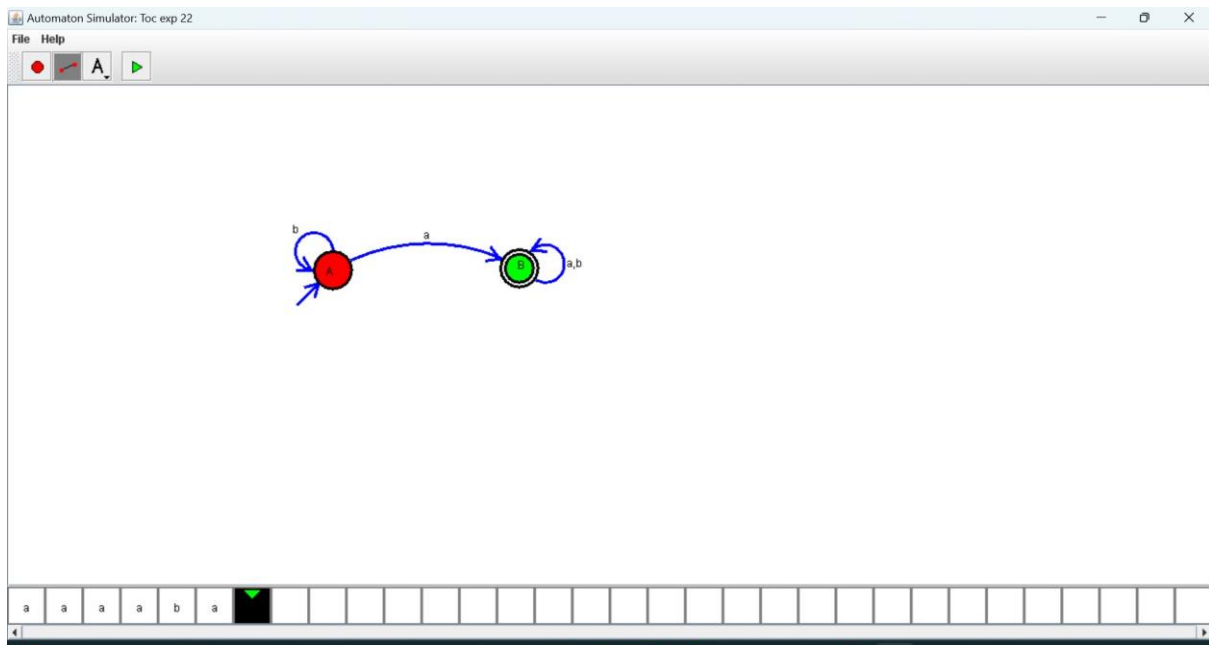
20. Design TM using simulator to perform subtraction of aaa-aa



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



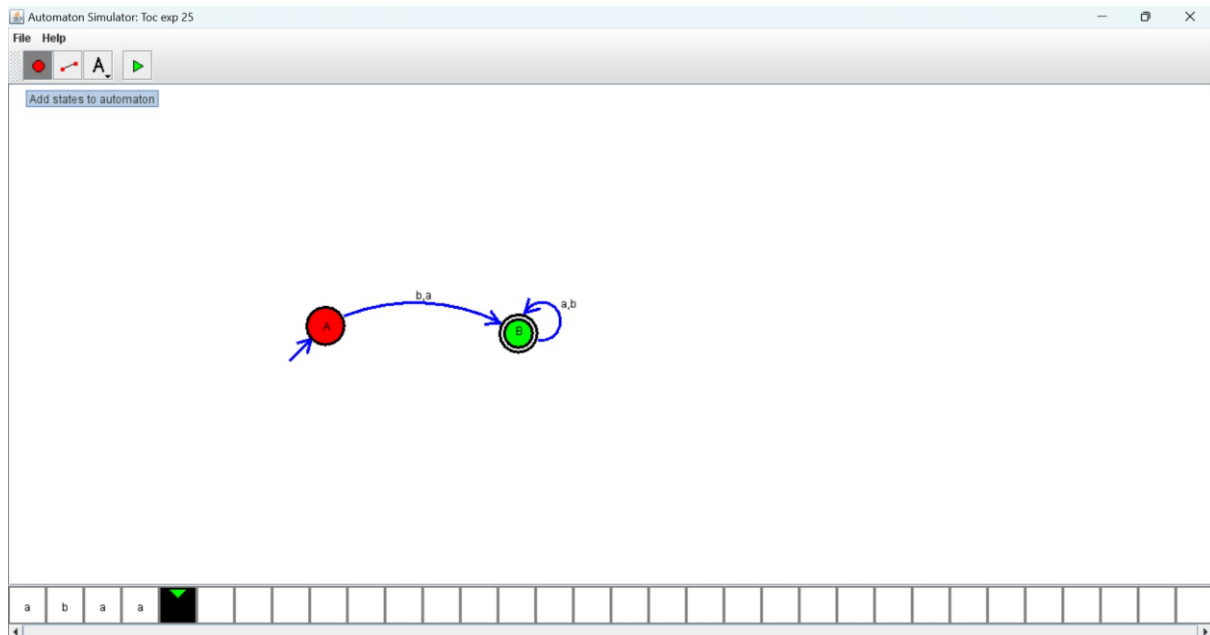
22. Design DFA using simulator to accept odd number of a's



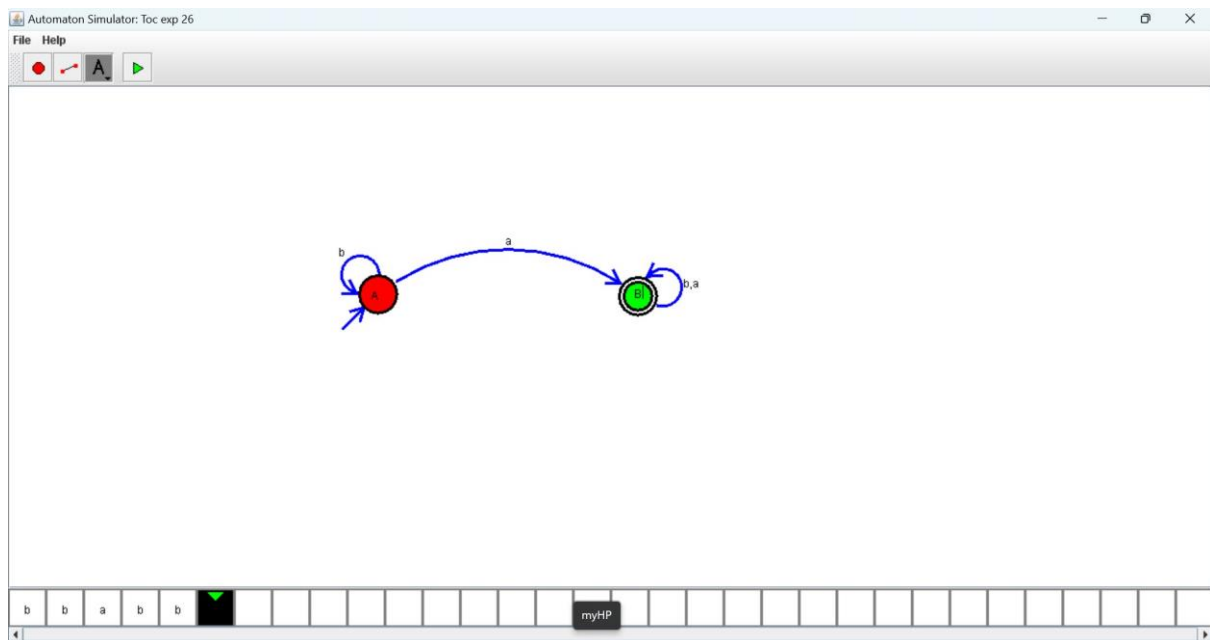
The screenshot shows the Automaton Simulator interface. The title bar reads "Automaton Simulator: Toc exp 23". The menu bar includes "File" and "Help". The "Help" menu is open, showing options: "Help...", "About...", and a green play button. The main workspace displays a finite state automaton with three states: a start state (red circle with a blue arrow), a middle state (red circle), and an accept state (green circle with a double border). Transitions are labeled 'a' and 'b'. The start state has a self-loop on 'a' and a transition to the middle state on 'b'. The middle state has a self-loop on 'b' and a transition to the accept state on 'a'. The accept state has a self-loop on 'b,a'. Below the workspace, an input sequence is shown in a row of boxes: "a", "a", "a", "b", "a", "b", followed by a green play button and several empty boxes.

The screenshot shows the 'Automaton simulator: 10c exp 24' window. The top toolbar includes 'File', 'Help', a red circle icon, a red arrow icon, a text input field with 'A', and a green play button. The main area displays a finite state automaton with three states: two red states and one green state. The first red state has a self-loop labeled 'b' and an incoming arrow. A transition labeled 'a' leads to the second red state, which has a self-loop labeled 'a'. A transition labeled 'b' leads from the second red state to the green state, which has a self-loop labeled 'a,b'. At the bottom, an input sequence is shown in a row of boxes: 'a', 'a', 'b', 'a', 'b', 'a', followed by a black box with a green triangle, and then several empty boxes.

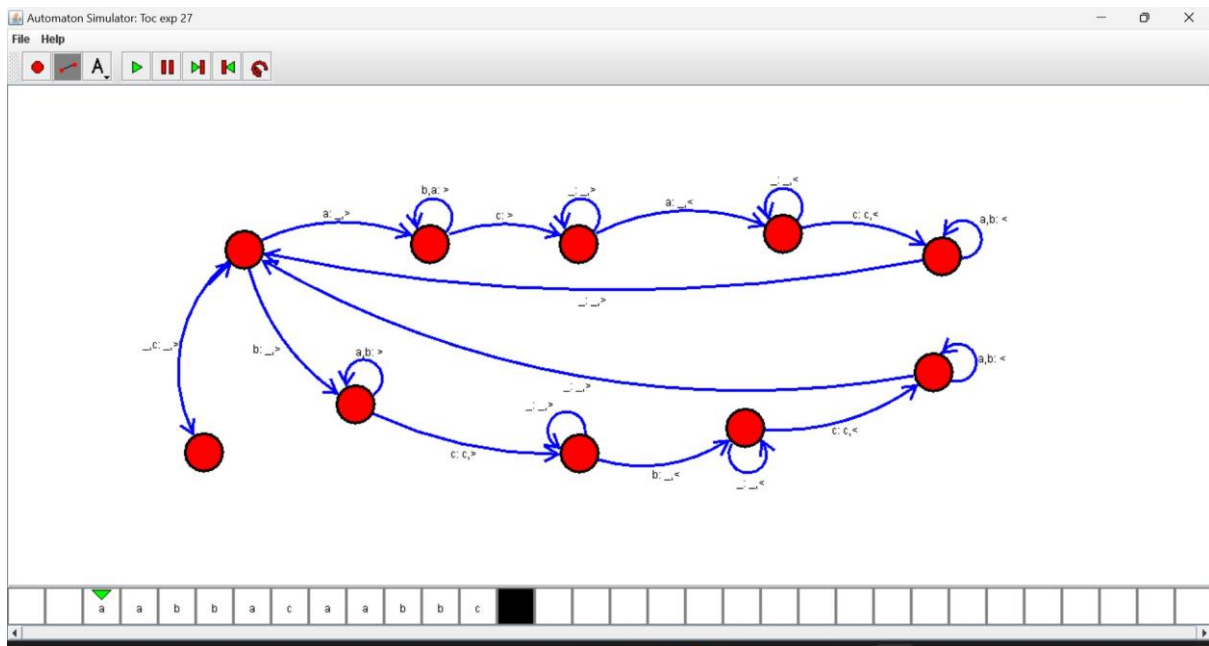
25. Design DFA using simulator to accept the string start with a or b over the set $\{a, b\}$



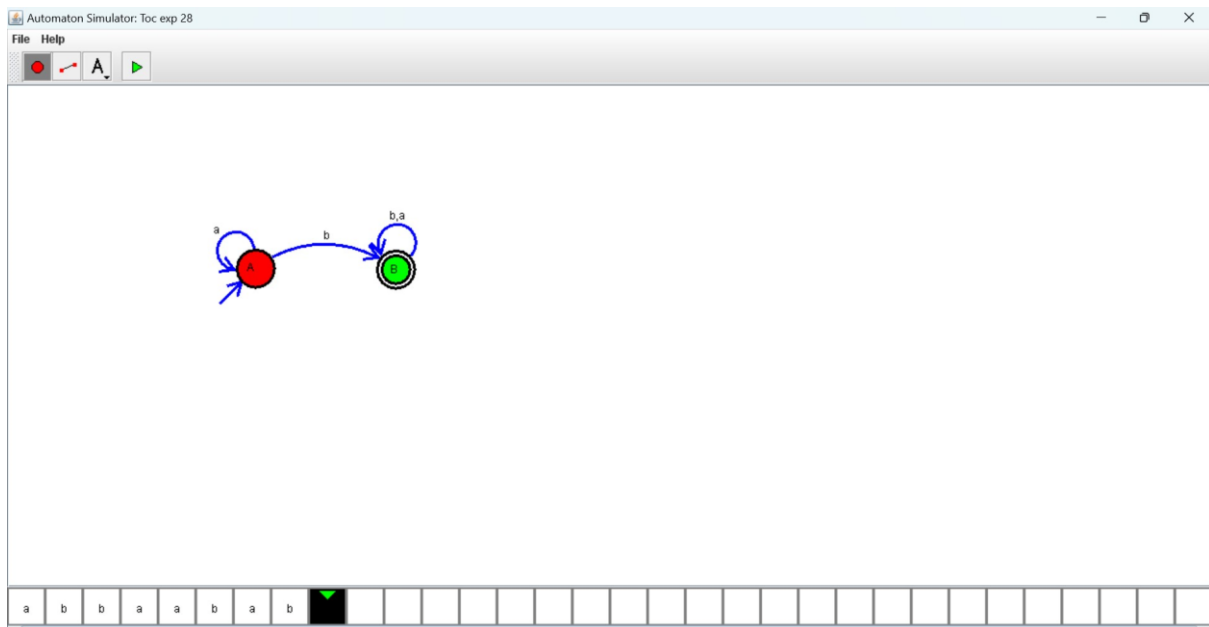
26. Design TM using simulator to accept the input string Palindrome bbabb



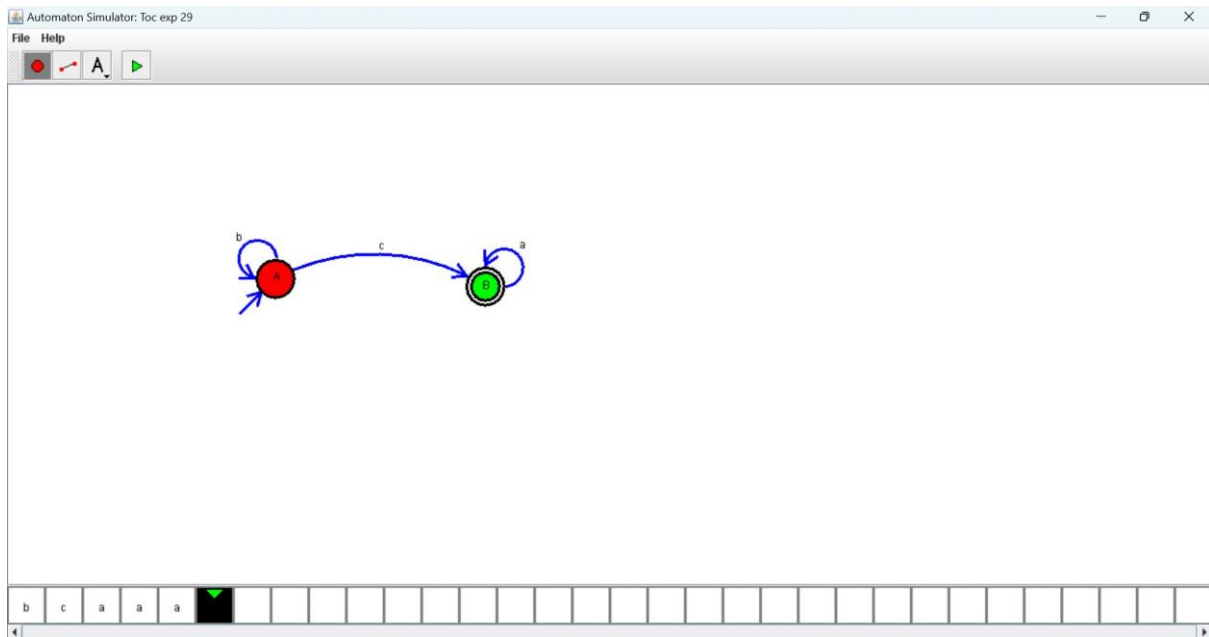
27. Design TM using simulator to accept the input string wcw



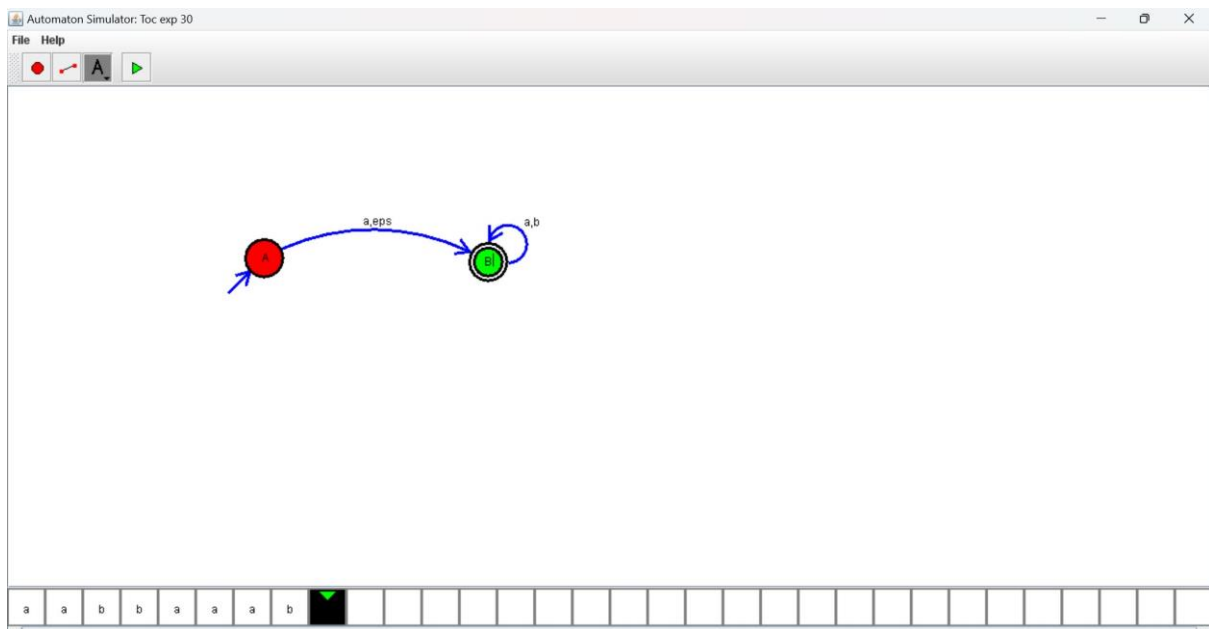
28. Design DFA using simulator to accept the string the end with ab over set {a, b} W= abbaabab



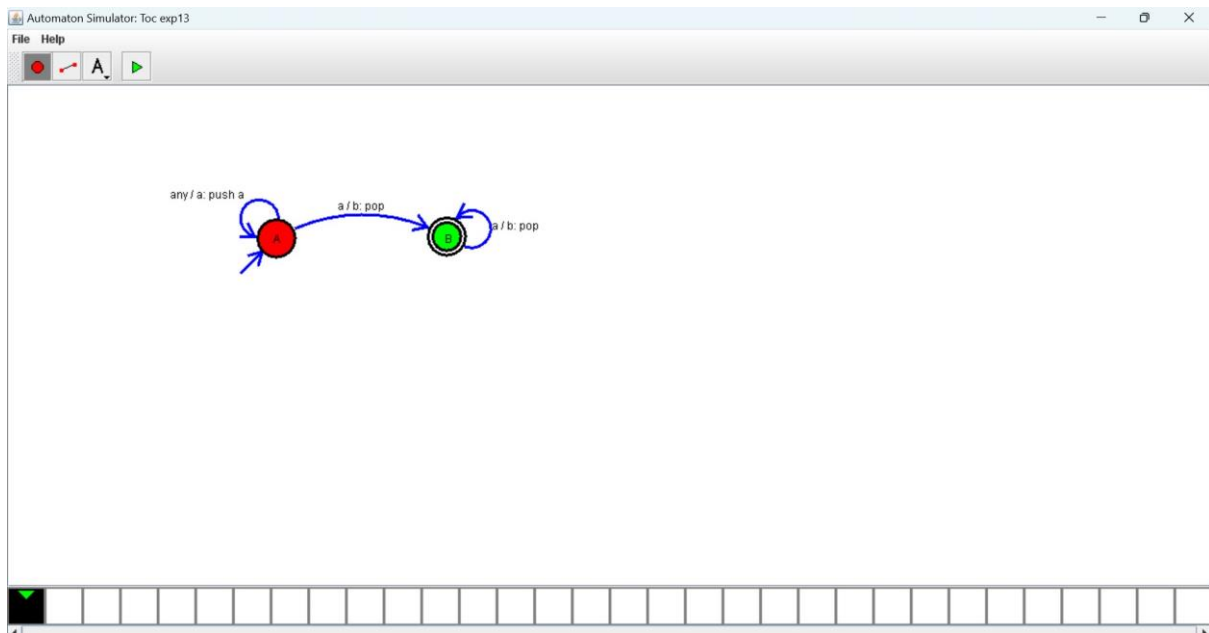
29. Design DFA using simulator to accept the input string “bc”, ” c”, and” bcaaa”.



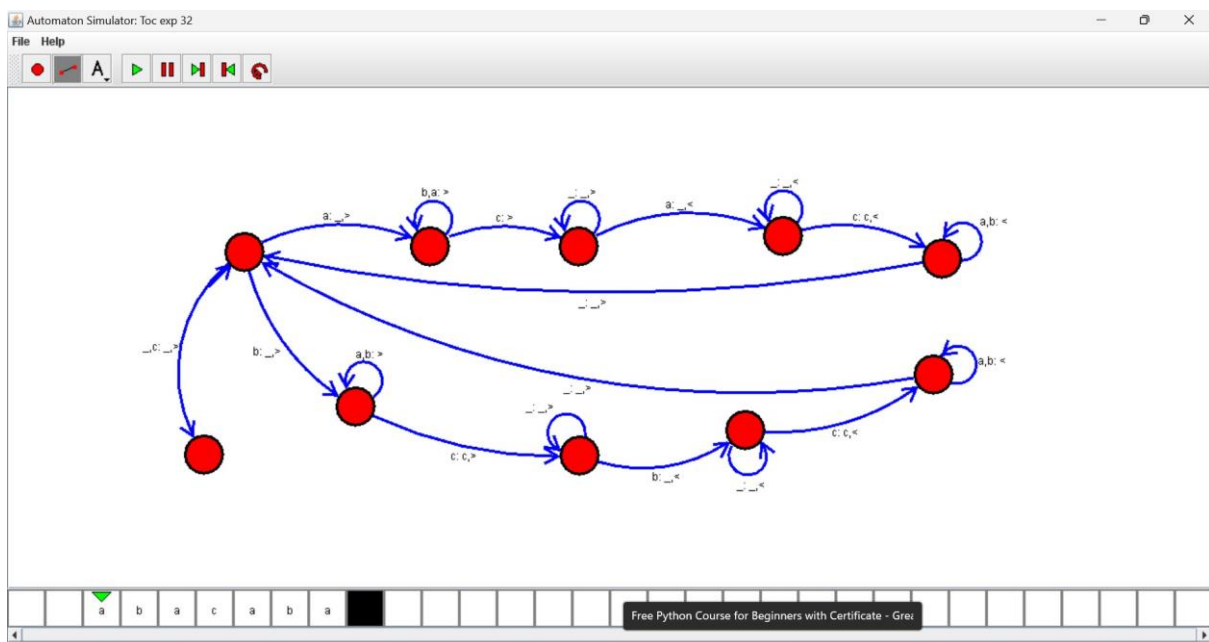
30. Design NFA to accept any number of a's where input = {a, b}.



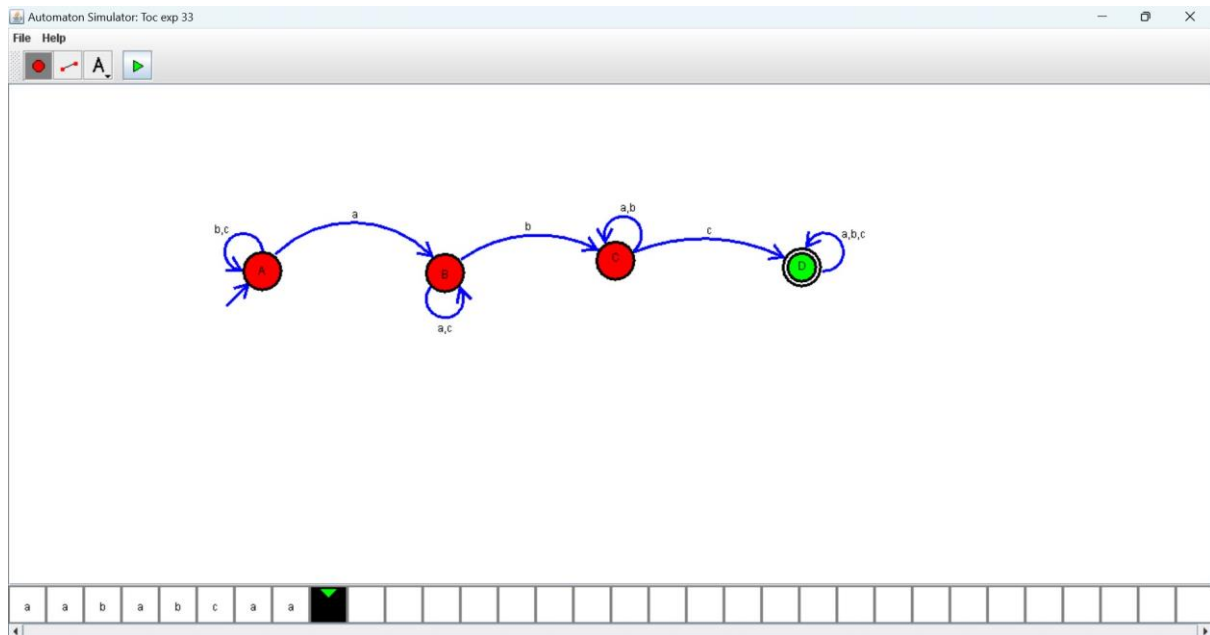
31. Design PDA using simulator to accept the input string $a^n b^n$



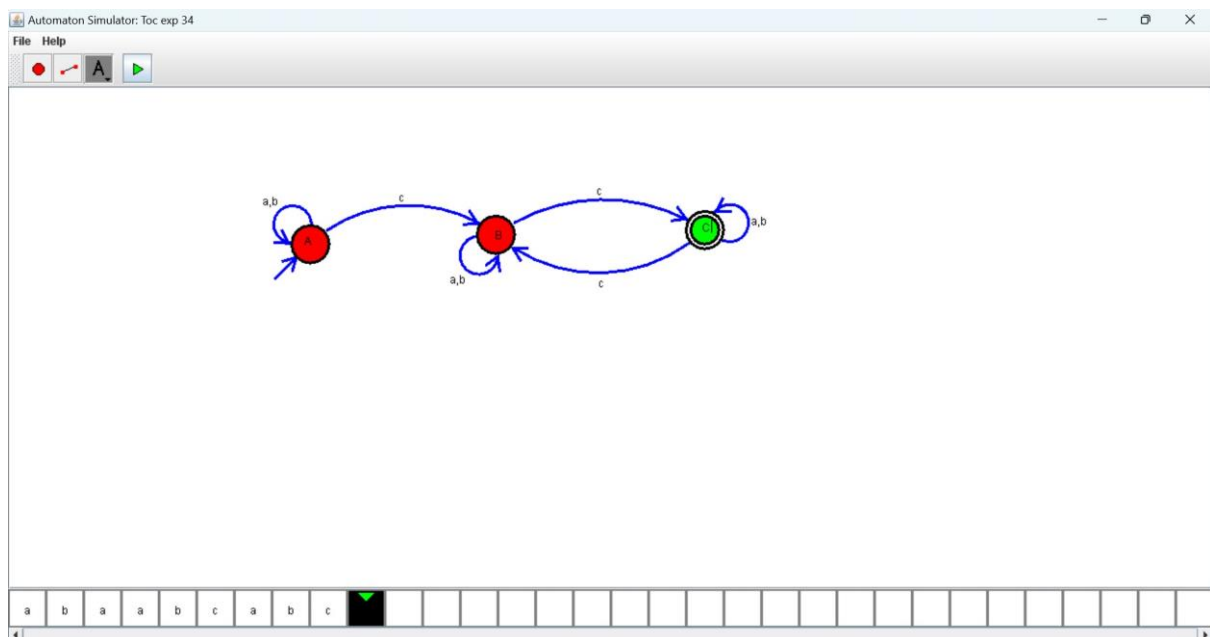
32. Design TM using simulator to perform string comparison where $w = \{aba\}$



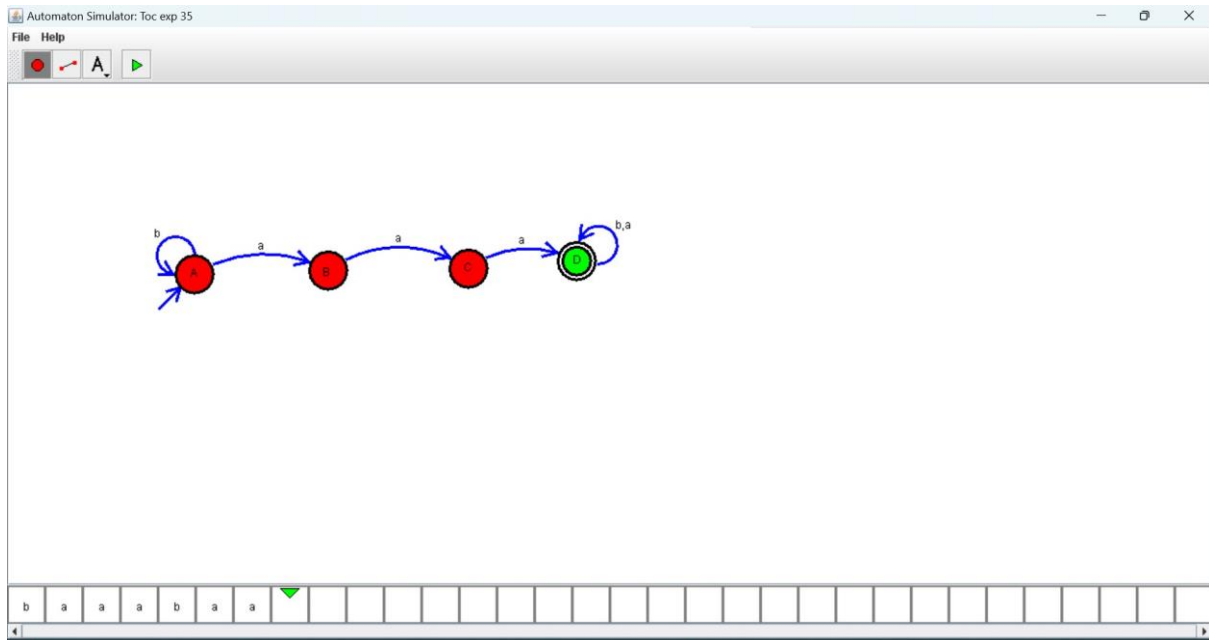
33. Design DFA using simulator to accept the string having 'abc' as substring over the set {a, b, c}



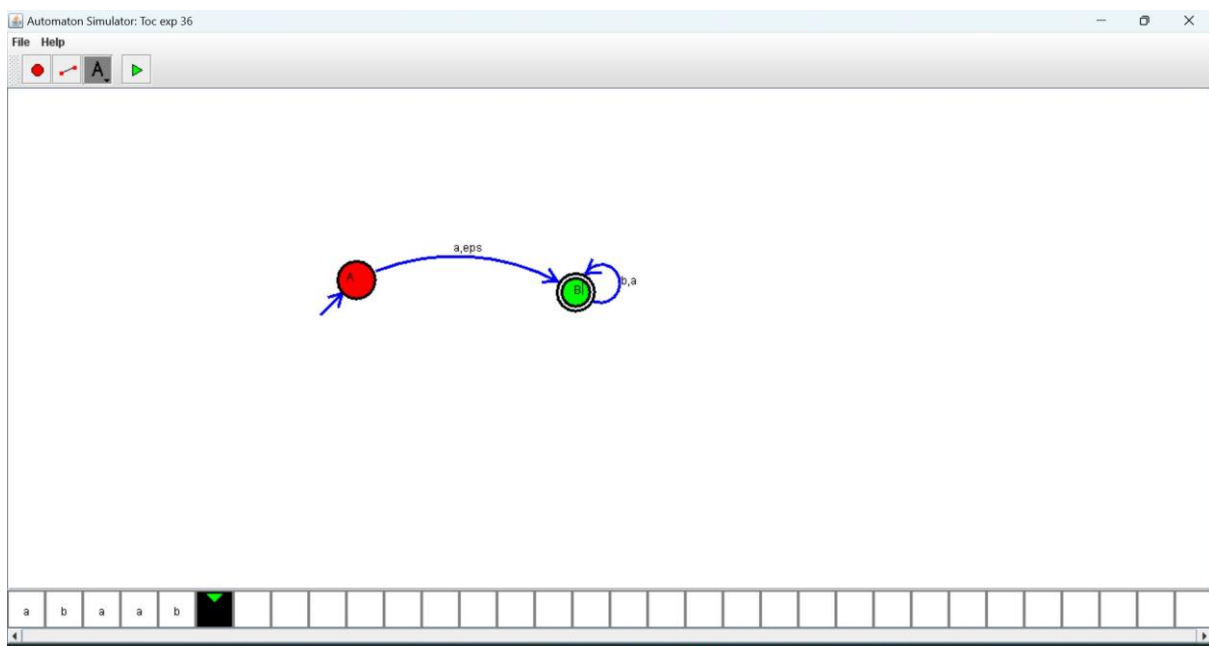
34. Design DFA using simulator to accept even number of c's over the set {a, b, c}



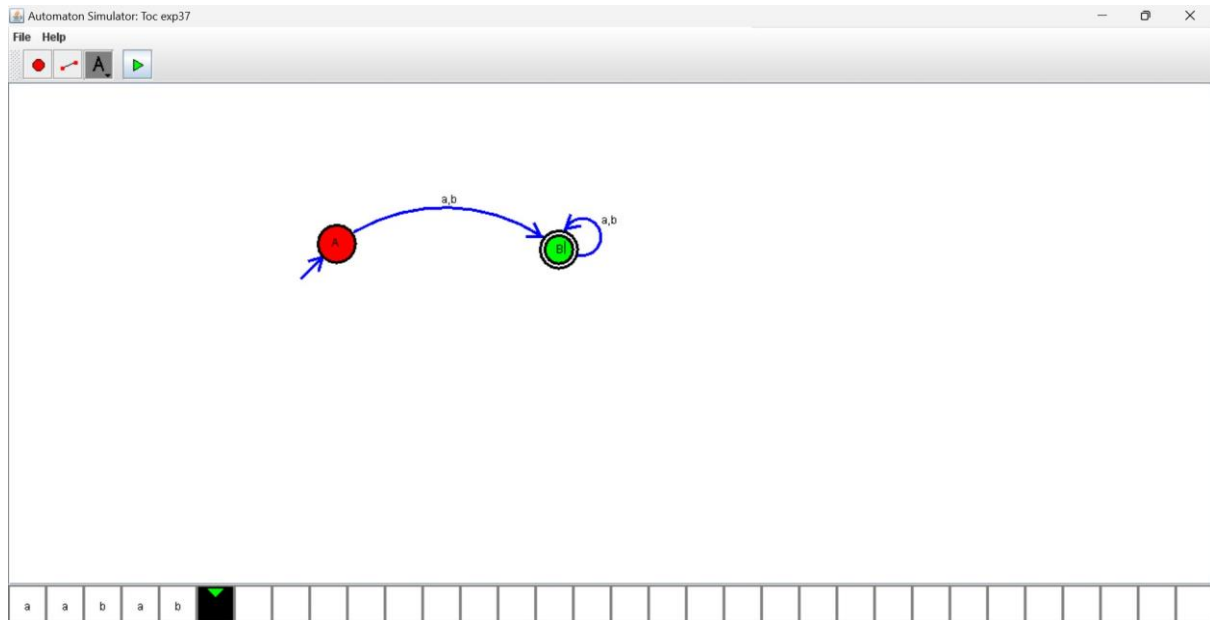
35. Design DFA using simulator to accept strings in which a's always appear tripled over input {a, b}



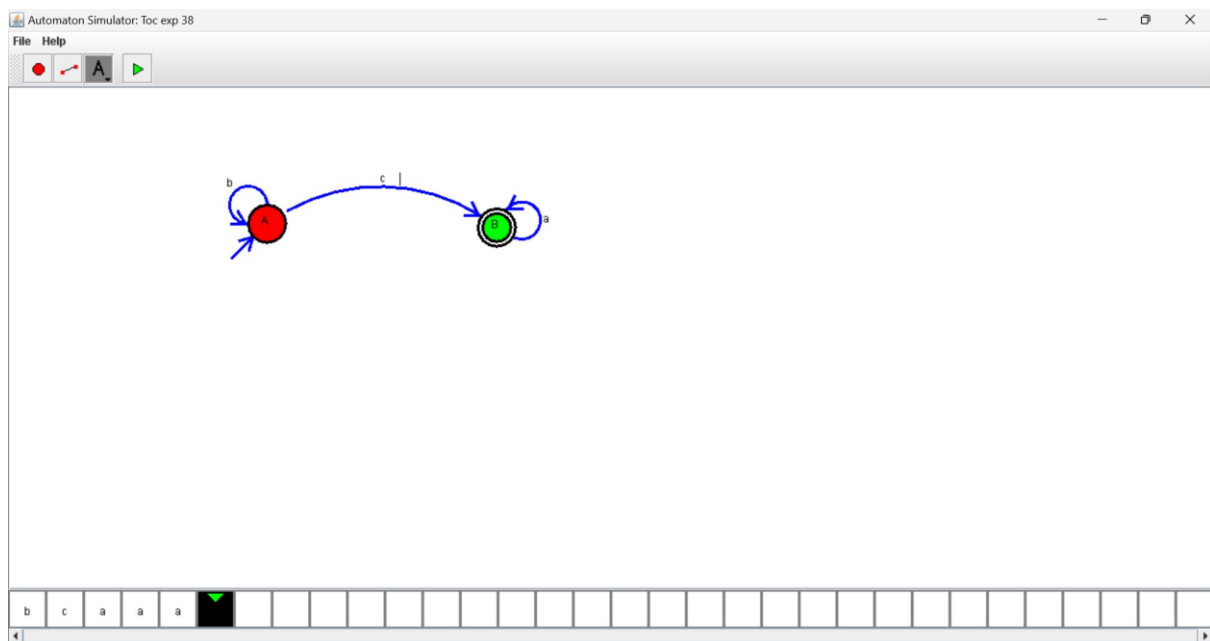
36. Design NFA using simulator to accept the string the start with a and end with b over set {a, b} and check W= abaab is accepted or not.



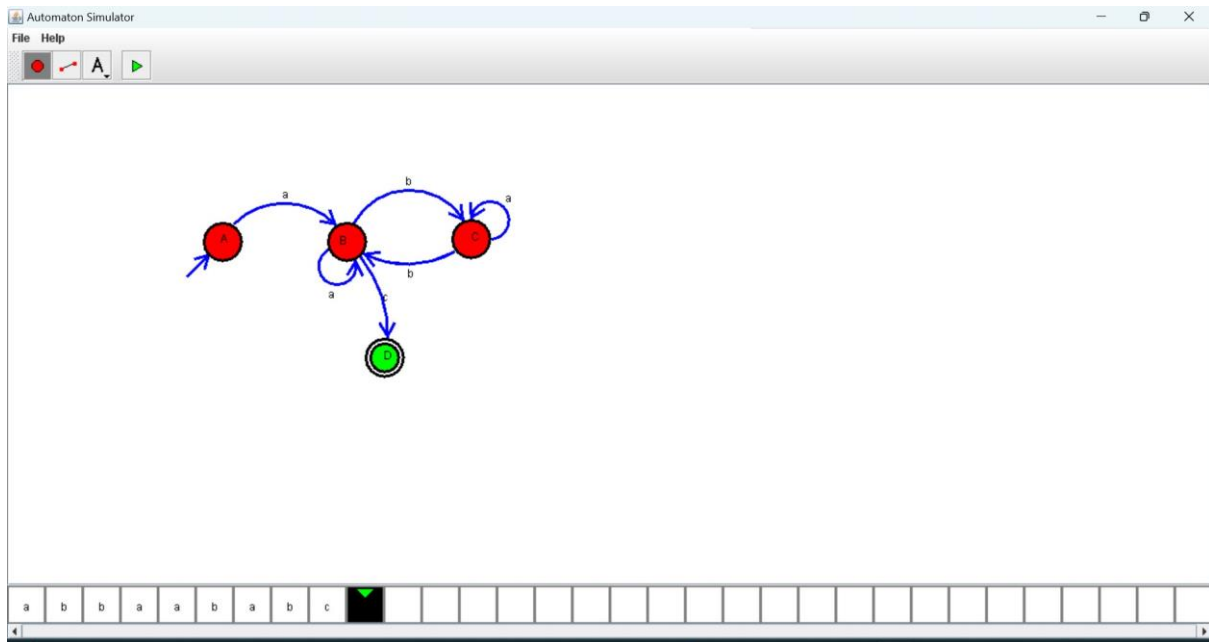
37. Design NFA using simulator to accept the string that start and end with different symbols over the input $\{a, b\}$.



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}.

