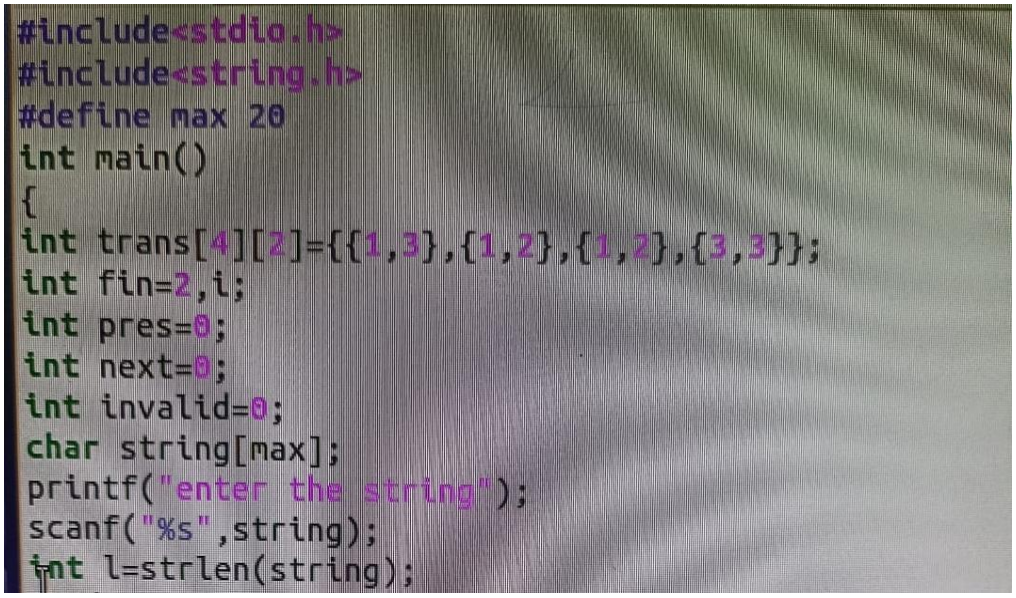
THEORY OF COMPUTATION-LAB

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

PROGRAM:



```
#include<stdio.h>
#include<string.h>
#define max 20
int main()
{
int trans[4][2]={{1,3},{1,2},{1,2},{3,3}};
int fin=2,i;
int pres=0;
int next=0;
int invalid=0;
char string[max];
printf("enter the string");
scanf("%s",string);
int l=strlen(string);
```

```
for(i=0;i<l;i++)
{
if(string[i]=='a')
next=trans[pres][0];
else if(string[i]=='b')
next=trans[pres][1];
else
invalid=1;
pres=next;
}
if(invalid==1)
{
printf("invalid input");
}
else if(pres=fin)
{
printf("accept\n");
}
else
printf("don't accept");
}
```

```
sse07@sse07-OptiPlex-330:~$ cd Desktop
sse07@sse07-OptiPlex-330:~/Desktop$ cc toc.c
sse07@sse07-OptiPlex-330:~/Desktop$ ./a.out
enter the stringabbbbabab
accept
sse07@sse07-OptiPlex-330:~/Desktop$
```

Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) S → 0A1 A → 0A | 1A | ε

main.c

```c
#include<stdio.h>
#include<string.h>
int main(){
char s[100];
int i,flag;
int l;
printf("enter a string to check:");
scanf("%s",s);
l=strlen(s);
flag=1;
for(i=0;i<l;i++)
{
if(s[i]!='0' && s[i]!='1')
{
 flag=0;
}
}
if(flag!=1)
printf("string is Not Valid\n");
if(flag==1)
{
if (s[0]=='0'&&s[l-1]=='1')
printf("string is accepted\n");
else
printf("string is Not accepted\n");
}
}
```

```
enter a string to check:011111101
string is accepted
```

Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) S → 0S0 | 1S1 | 0 | 1 | ε

```c
#include<stdio.h>
#include<string.h>
void main()
{
char s[100];
int i,flag,flag1,a,b;
int l;
printf("enter a string to check:");
scanf("%s",s);
l=strlen(s);
flag=1;
for(i=0;i<l;i++)
{
if(s[i]!='0' && s[i]!='1')
{
flag=0;
}
}
if(flag!=1)
printf("string is Not Valid\n");
if(flag==1)
{
flag1=1;
a=0;b=l-1;
while(a!=(l/2))
{
if(s[a]!=s[b])
{
flag1=0;
}
a=a+1;
b=b-1;
}
if (flag1==1)
{
 printf("The string is a palindrome\n");
 printf("string is accepted\n");
}
```

```
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");
}
}
}
```

```
he string is a palindrome
tring is accepted
```

Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) S → 0S0 | A A → 1A | ε

```
#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') // Count the no of 0s in the front
{
count1++;
i++;
```

```
}
while(s[i]=='1')
{
i++; // Skip all 1s
}
flag1=1;
count2=0;
while(i<l)
{
if(s[i]=='0')// Count the no of 0s at the end
{
count2++;
}
else
{
flag1=0;
}
i++;
}
if(flag1==1)
{
if(count1==count2)
 {
 printf("The string satisfies the condition 0n1m0n\n");
 printf("String Accepted\n");
 }
 else
 {
 printf("The string does not satisfy the condition 0n1m0n\n");
 printf("String Not Accepted\n");
 }
}
else
 {
 printf("The string does not satisfy the condition 0n1m0n\n");
 printf("String Not Accepted\n");
}
```

input

```
enter a string to check:00011000
The string satisfies the condition 0n1m0n
String Accepted
```

Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) S → 0S1 | ε

```c
#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) // If string length is odd
 {
 printf("The string does not satisfy the condition 0n1n\n");
 printf("String Not Accepted\n");
 }
else
{
// To check first half contains 0s
flag1=1;
for(i=0;i<(l/2);i++)
{
if(s[i]!='0')
{
flag1=0;
}
```

```
flag1=1;
for(i=0;i<(l/2);i++)
{
if(s[i]!='0')
{
flag1=0;
}
}
// To check second half contains 1s
flag2=1;
for(i=l/2;i<l;i++)
{
if(s[i]!='1')
{
flag2=0;
}
}
if(flag1==1 && flag2==1)
 {
 printf("The string satisfies the condition 0n1n\n");
 printf("String Accepted\n");
 }
else
 {
 printf("The string does not satisfy the condition 0n1n\n");
 printf("String Not Accepted\n");
}
}
}
}
```

```
enter a string to check:000111
The string satisfies the condition 0n1n
String Accepted
```

Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG) S → A101A A → 0A | 1A | ε

```c
#include<stdio.h>
#include<string.h>
int main()
{
char s[100];
int i,flag,flag1;
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 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;
}
}
}
```

```
33  printf("Substring 101 exists. String accepted\n");
34  break;
35  }
36  }
37  }
38  if(flag1==0)
39  printf("Substring 101 does not exist. String not accepted\n");
40  }
41  }
42
```

input

```
nter a string to check:11101111
tring is Valid
ubstring 101 exists. String accepted
```

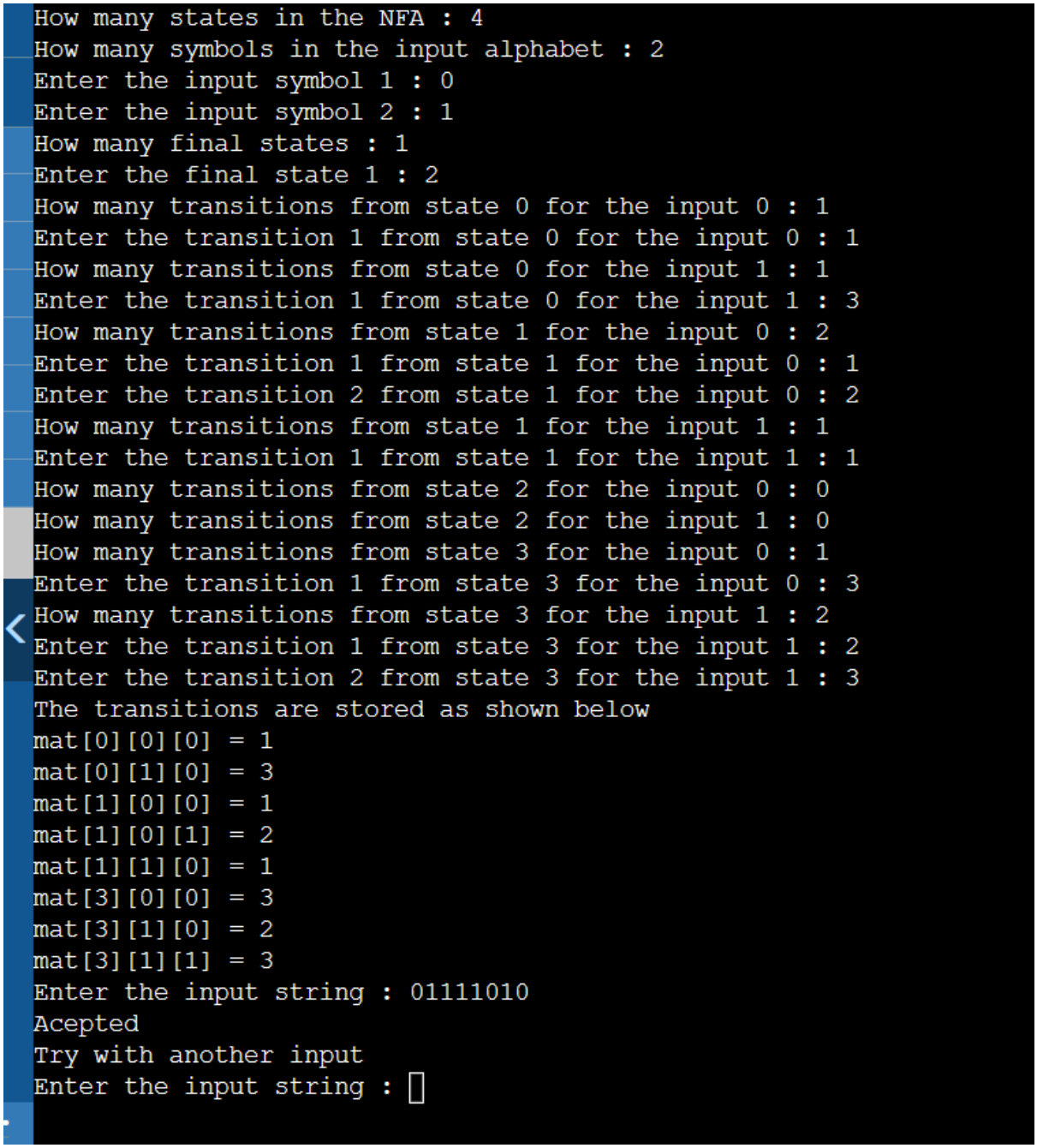
Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with o and end with 1

```c
#include<stdio.h>
#include<string.h>
int main()
{
int i,j,k,l,m,next_state[20],n,mat[10][10][10],flag,p;
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]);
}
//Initialize all entries with -1 in Transition table
for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
for(k=0;k<10;k++)
{
mat[i][j][k]=-1;
}
}
}
//Get input from the user and fill the 3D transition table
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
```

```
 else
 {
 printf("Invalid input\n");
 }
 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;
}
```

```
}
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("Acepted\n");
else
printf("Not accepted\n");
printf("Try with another input\n");
}
}
```



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

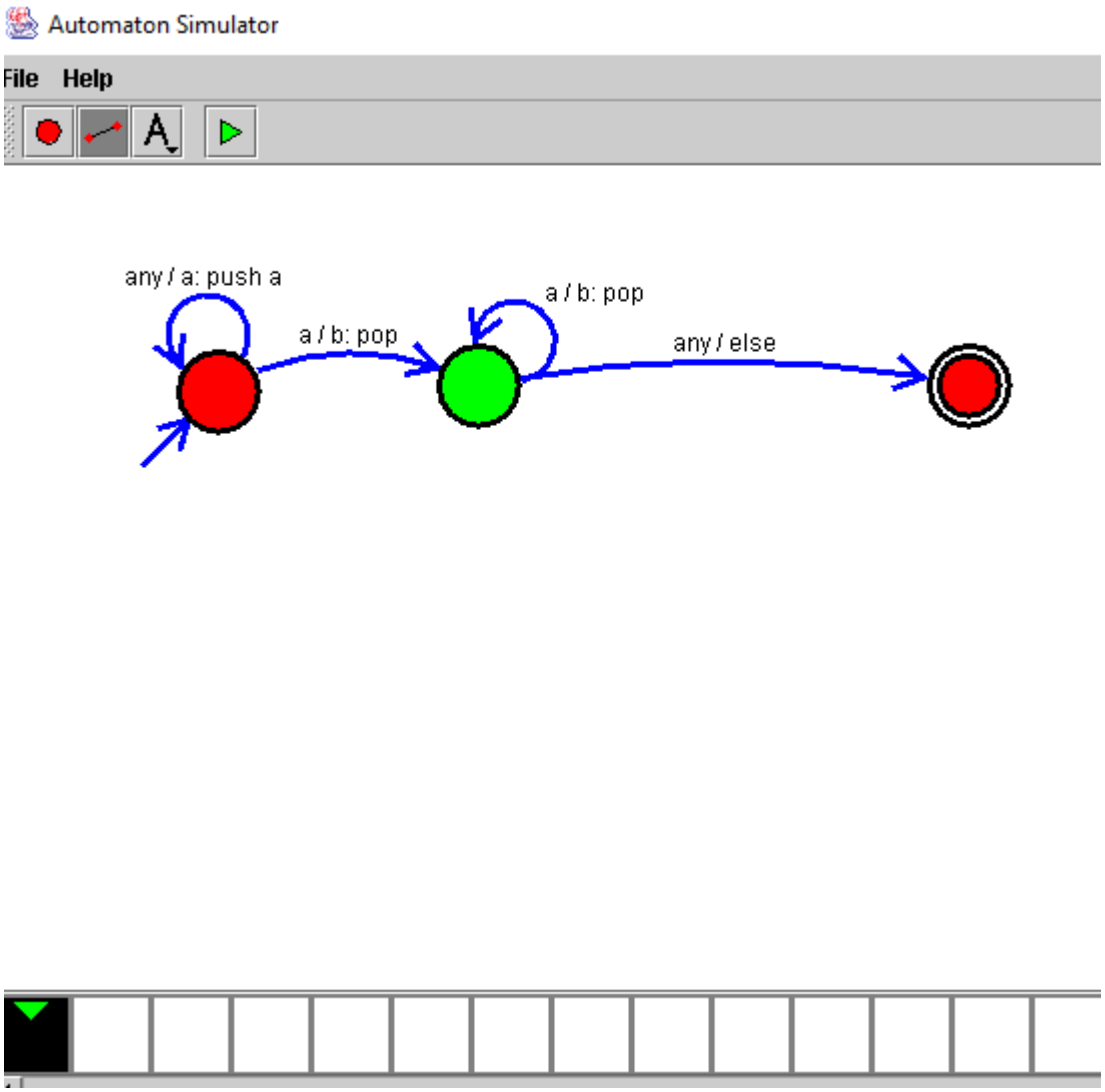
```c
#include<stdio.h>
#include<string.h>
int trans_table[10][5][3];
char symbol[5],a;
int e_closure[10][10],ptr,state;
void find_e_closure(int x);
int main()
{
int i,j,k,n,num_states,num_symbols;
for(i=0;i<10;i++)
{
for(j=0;j<5;j++)
{
for(k=0;k<3;k++)
{
trans_table[i][j][k]=-1;
}
}
}
printf("How may states in the NFA with e-moves:");
scanf("%d",&num_states);
printf("How many symbols in the input alphabet including e :");
scanf("%d",&num_symbols);
printf("Enter the symbols without space. Give 'e' first:");
scanf("%s",symbol);
for(i=0;i<num_states;i++)
{
for(j=0;j<num_symbols;j++)
{
printf("How many transitions from state %d for the input %c:",i,symbol[j]);
scanf("%d",&n);
for(k=0;k<n;k++)
{
printf("Enter the transitions %d from state %d for the input %c :", k+1,i,symbol[j]);
scanf("%d",&trans_table[i][j][k]);
}
}
}
}
```

```
for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
e_closure[i][j]=-1;
}
}
for(i=0;i<num_states;i++)
e_closure[i][0]=i;
for(i=0;i<num_states;i++)
{
if(trans_table[i][0][0]==-1)
continue;
else
{
state=i;
ptr=1;
find_e_closure(i);
}
}
for(i=0;i<num_states;i++)
{
printf("e-closure(%d)= {",i);
for(j=0;j<num_states;j++)
{
if(e_closure[i][j]!=-1)
{
printf("%d, ",e_closure[i][j]);
}
}
printf("}\n");
}
}
void find_e_closure(int x)
{
int i,j,y[10],num_trans;
i=0;
while(trans_table[x][0][i]!=-1)
```

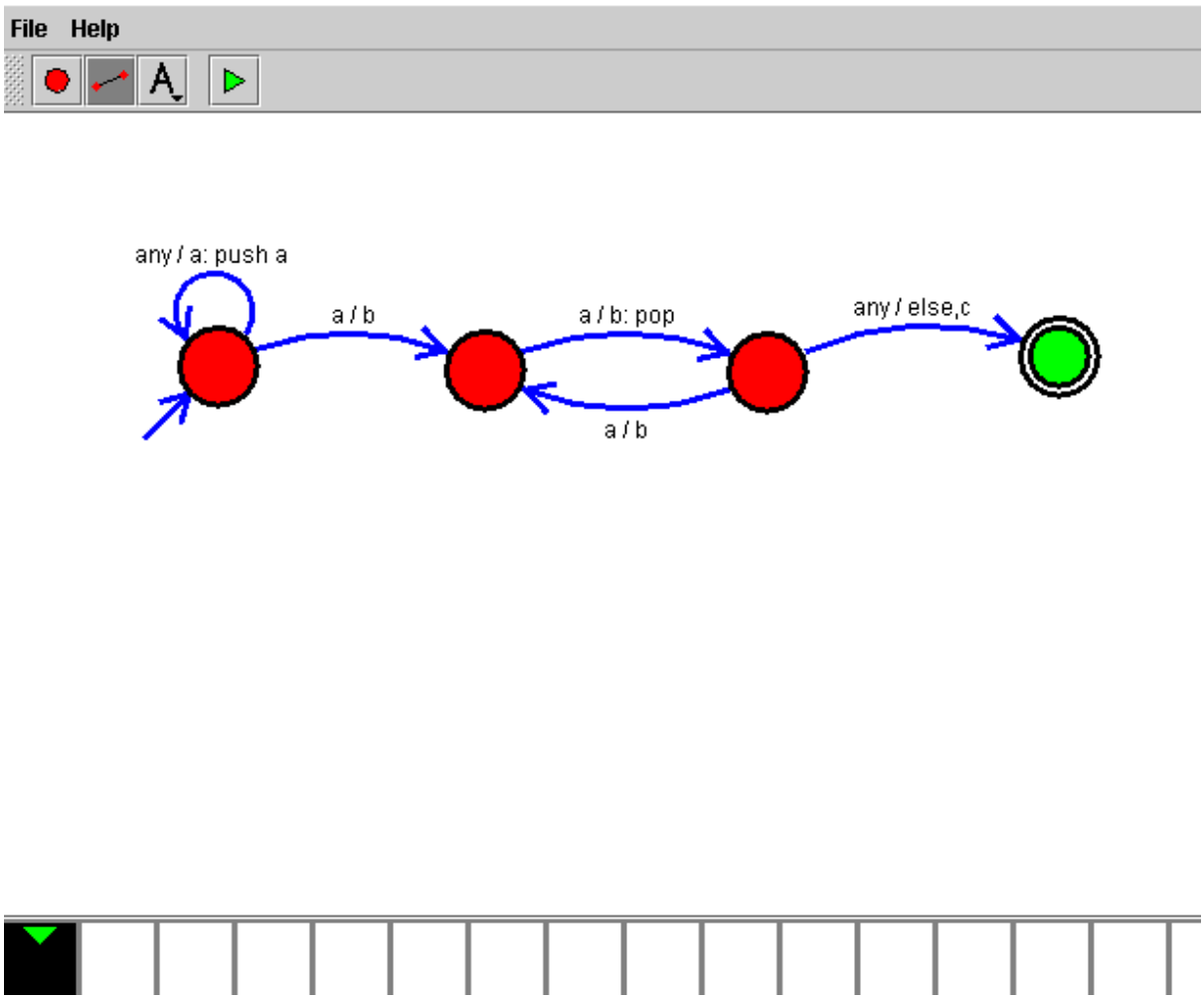
```
i=0;
while(trans_table[x][0][i]!=-1)
{
y[i]=trans_table[x][0][i];
i=i+1;
}
num_trans=i;
for(j=0;j<num_trans;j++)
{
e_closure[state][ptr]=y[j];
ptr++;
find_e_closure(y[j]);
}
}
```

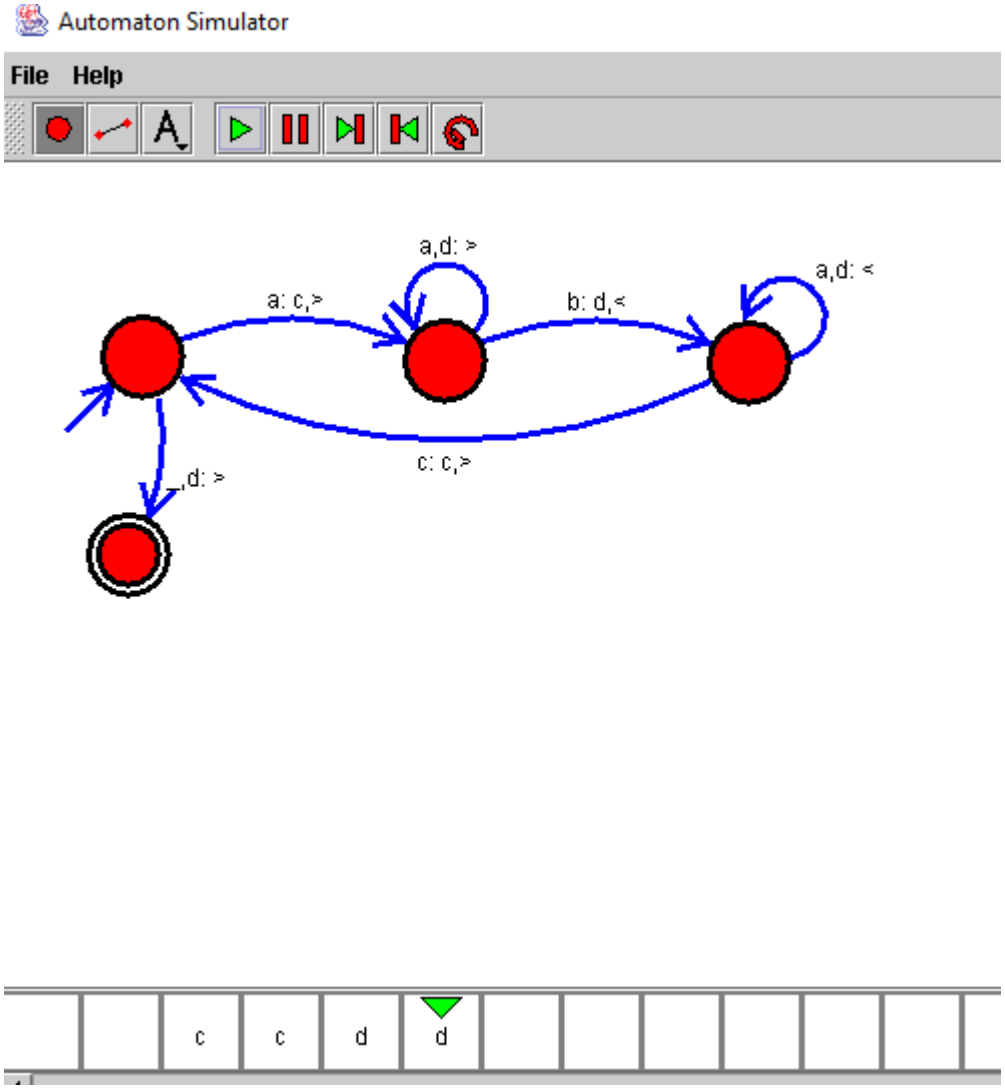
```
How many symbols in the input alphabet including e :3
Enter the symbols without space. Give 'e' first:e01
How many transitions from state 0 for the input e:1
Enter the transitions 1 from state 0 for the input e :1
How many transitions from state 0 for the input 0:0
How many transitions from state 0 for the input 1:1
Enter the transitions 1 from state 0 for the input 1 :1
How many transitions from state 1 for the input e:1
Enter the transitions 1 from state 1 for the input e :2
How many transitions from state 1 for the input 0:2
Enter the transitions 1 from state 1 for the input 0 :0
Enter the transitions 2 from state 1 for the input 0 :1
How many transitions from state 1 for the input 1:0
How many transitions from state 2 for the input e:0
How many transitions from state 2 for the input 0:0
How many transitions from state 2 for the input 1:0
e-closure(0)= {0, 1, 2, }
e-closure(1)= {1, 2, }
e-closure(2)= {2, }
```

Design PDA using simulator to accept the input string aabb

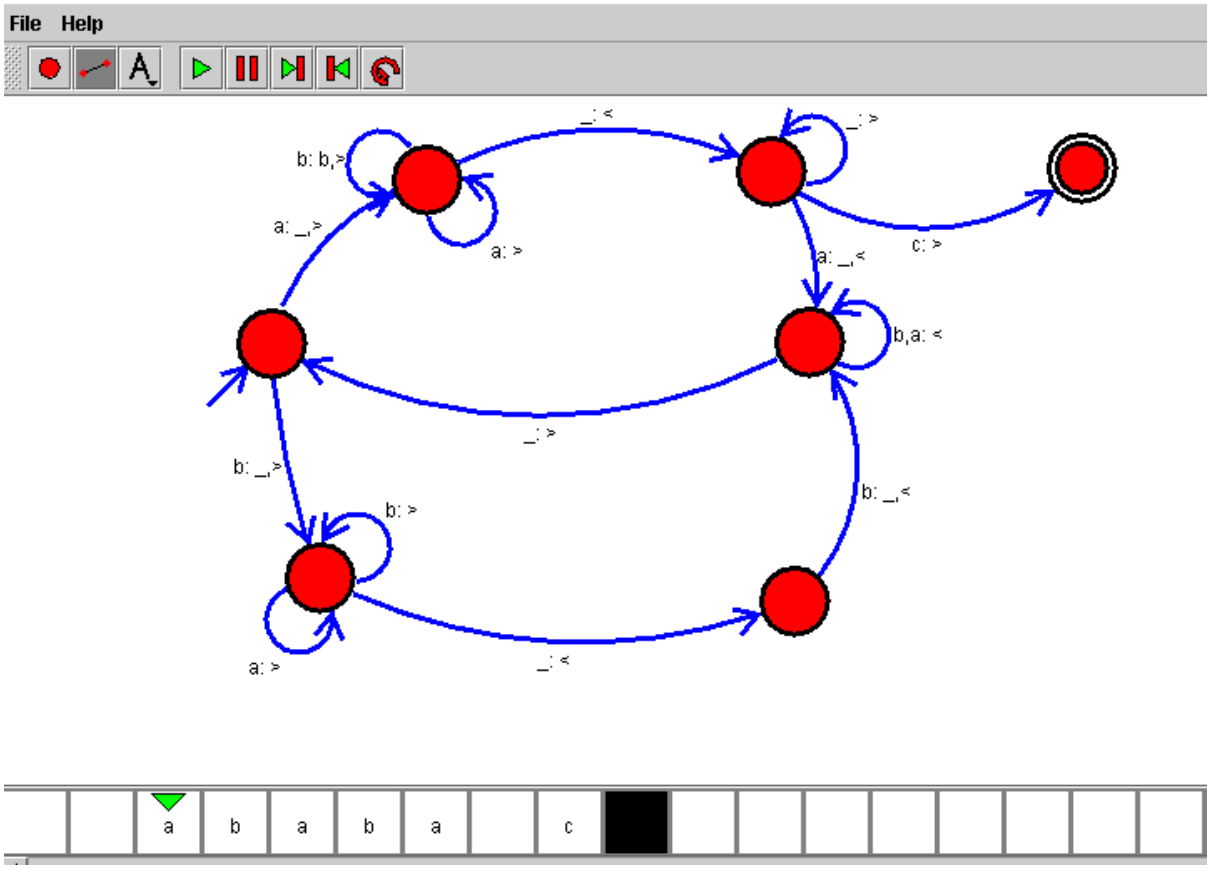


14. Design PDA using simulator to accept the input string a^nb^2n

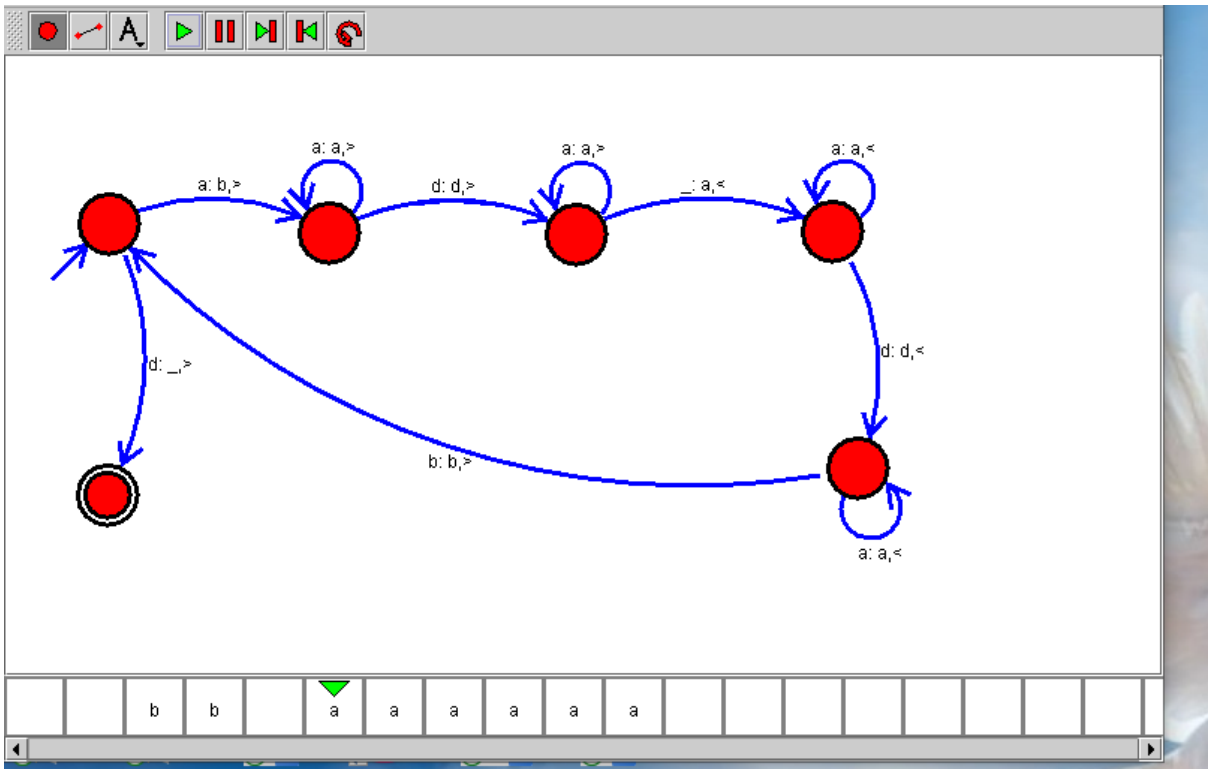

File Help
any / a: push a
a / b
a / b: pop
any / else,c
a / b



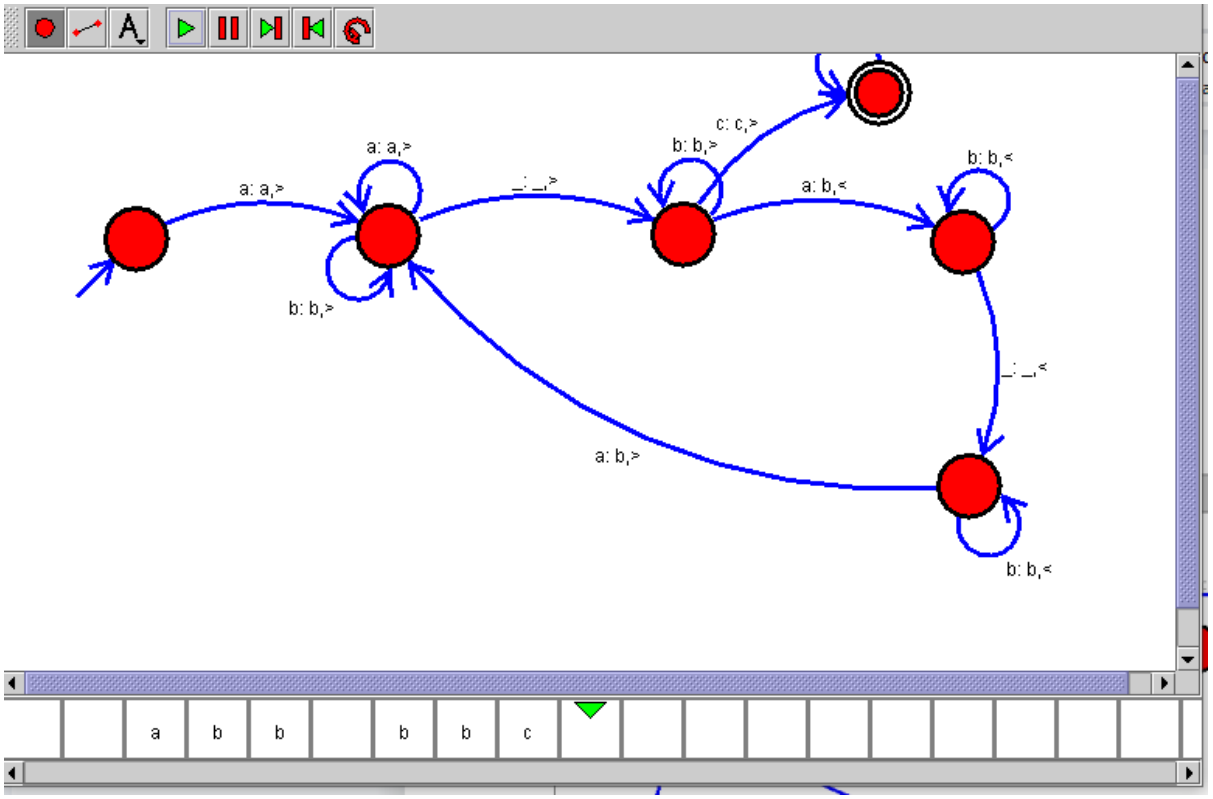
Design TM using simulator to accept the input string Palindrome ababa



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



Design TM using simulator to perform subtraction of aaa-aa


a: a,>
a: a,>
b: b,>
_: _,>
c: c,>
b: b,>
a: b,<
b: b,<
_: _,<
a: b,>
b: b,<
a
b
b
b
b
c