

# CSA1354 – Theory of Computation with Productions

## Lab Experiments – Slot A

**Name :- Aswini . P**

**Reg.no :- 192011399**

**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>
#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=1;

        present_state=next_state;
    }

    if(invalid==1)
    {

        printf("Invalid input");

    }

    else if(present_state==final_state)

        printf("Accept\n");

    else

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

}

```

### Output :-

```

1  #include<stdio.h>
2  #include<string.h>
3  #define max 20
4  int main()
5  {
6      int trans_table[4][2]={{1,3},{1,2},{1,2},{3,3}};
7      int final_state=2,i;
8      int present_state=0;
9      int next_state=0;
10     int invalid=0;
11     char input_string[max];
12     printf("Enter the string:");
13     scanf("%s",input_string);
14     int l=strlen(input_string);
15     for(i=0;i<l;i++)
16     {
17         if(input_string[i]=='a')
18             next_state=trans_table[present_state][0];
19         else if(input_string[i]=='b')
20             next_state=trans_table[present_state][1];
21         else
22             invalid=1;
23         present_state=next_state;
24     }
25     if(invalid==1)
26     {
27         printf("Invalid input");
28     }
29     else if(present_state==final_state)
30         printf("Accept\n");
31     else
32         printf("Don't Accept\n");
33 }

```

```

C:\Users\pavan\OneDrive\Do... x + v
Enter the string:abbaba
Don't Accept

Process returned 0 (0x0)   execution time : 17.956 s
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()

```

```

{
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]);
}
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(0);
}
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 :-

<pre> 1  #include&lt;stdio.h&gt; 2  #include&lt;string.h&gt; 3  int main() 4  { 5      int i,j,k,l,m,next_state[20],n,mat[10][10][10],flag,p; 6      int num_states,final_state[5],num_symbols,num_final; 7      int present_state[20],prev_trans,new_trans; 8      char ch,input[20]; 9      int symbol[5],inp,inpl; 10     printf("How many states in the NFA : "); 11     scanf("%d",&amp;num_states); 12     printf("How many symbols in the input alphabet : "); 13     scanf("%d",&amp;num_symbols); 14     for(i=0;i&lt;num_symbols;i++) 15     { 16         printf("Enter the input symbol %d : ",i+1); 17         scanf("%d",&amp;symbol[i]); 18     } 19     printf("How many final states : "); 20     scanf("%d",&amp;num_final); 21     for(i=0;i&lt;num_final;i++) 22     { 23         printf("Enter the final state %d : ",i+1); 24         scanf("%d",&amp;final_state[i]); 25     } 26     for(i=0;i&lt;10;i++) 27     { 28         for(j=0;j&lt;10;j++) 29         { 30             for(k=0;k&lt;10;k++) 31             { 32                 mat[i][j][k]=-1; 33             } 34         } 35     } 36     for(i=0;i&lt;num_states;i++) 37     { 38         for(j=0;j&lt;num_symbols;j++) 39         { 40             printf("How many transitions from state %d for the input %d : ",i,symbol[j]); 41             scanf("%d",&amp;n); 42             for(k=0;k&lt;n;k++) 43             { 44                 printf("Enter the transition %d from state %d for the input %d : ",k+1,i,symbol[j]) </pre>	<pre> How many states in the NFA : 4 How many symbols in the input alphabet : 2 Enter the input symbol 1 : 0 Enter the input symbol 2 : 1 How many final states : 1 Enter the final state 1 : 2 How many transitions from state 0 for the input 0 : 1 Enter the transition 1 from state 0 for the input 0 : 1 How many transitions from state 0 for the input 1 : 1 Enter the transition 1 from state 0 for the input 1 : 3 How many transitions from state 1 for the input 0 : 2 Enter the transition 1 from state 1 for the input 0 : 1 Enter the transition 2 from state 1 for the input 0 : 2 How many transitions from state 1 for the input 1 : 1 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 shown 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 : 100100 Not accepted </pre>
---	--

**3. 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 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 :", 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)

{

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 :-

```

1  #include<stdio.h>
2  #include<string.h>
3  int trans_table[10][5][3];
4  char symbol[5],a;
5  int e_closure[10][10],ptr,state;
6  void find_e_closure(int x);
7  int main()
8  {
9      int i,j,k,n,num_states,num_symbols;
10     for(i=0;i<10;i++)
11     {
12         for(j=0;j<5;j++)
13         {
14             for(k=0;k<3;k++)
15             {
16                 trans_table[i][j][k]=-1;
17             }
18         }
19     }
20     printf("How many states in the NFA with e-moves:");
21     scanf("%d",&num_states);
22     printf("How many symbols in the input alphabet including e :");
23     scanf("%d",&num_symbols);
24     printf("Enter the symbols without space. Give 'e' first:");
25     scanf("%s",symbol);
26     for(i=0;i<num_states;i++)
27     {
28         for(j=0;j<num_symbols;j++)
29         {
30             printf("How many transitions from state %d for the input%c:",i,symbol[j]);
31             scanf("%d",&n);
32             for(k=0;k<n;k++)
33             {
34                 printf("Enter the transitions %d from state %d for the input%c :", k+1,i,symbol[j]);
35                 scanf("%d",&trans_table[i][j][k]);
36             }
37         }
38     }
39     for(i=0;i<10;i++)
40     {
41         for(j=0;j<10;j++)
42         {
43             e_closure[i][j]=-1;
44         }
45     }

```

```

How may states in the NFA with e-moves:3
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 input0:0
How many transitions from state 0 for the input1:1
Enter the transitions 1 from state 0 for the input1 :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 input0:2
Enter the transitions 1 from state 1 for the input0 :0
Enter the transitions 2 from state 1 for the input0 :1
How many transitions from state 1 for the input1:0
How many transitions from state 2 for the input:e:0
How many transitions from state 2 for the input0:0
How many transitions from state 2 for the input1:0
e-closure(0)= {0, 1, 2, }
e-closure(1)= {1, 2, }
e-closure(2)= {2, }

Process returned 0 (0x0)   execution time : 47.323 s
Press any key to continue.

```

**4. 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 :-**

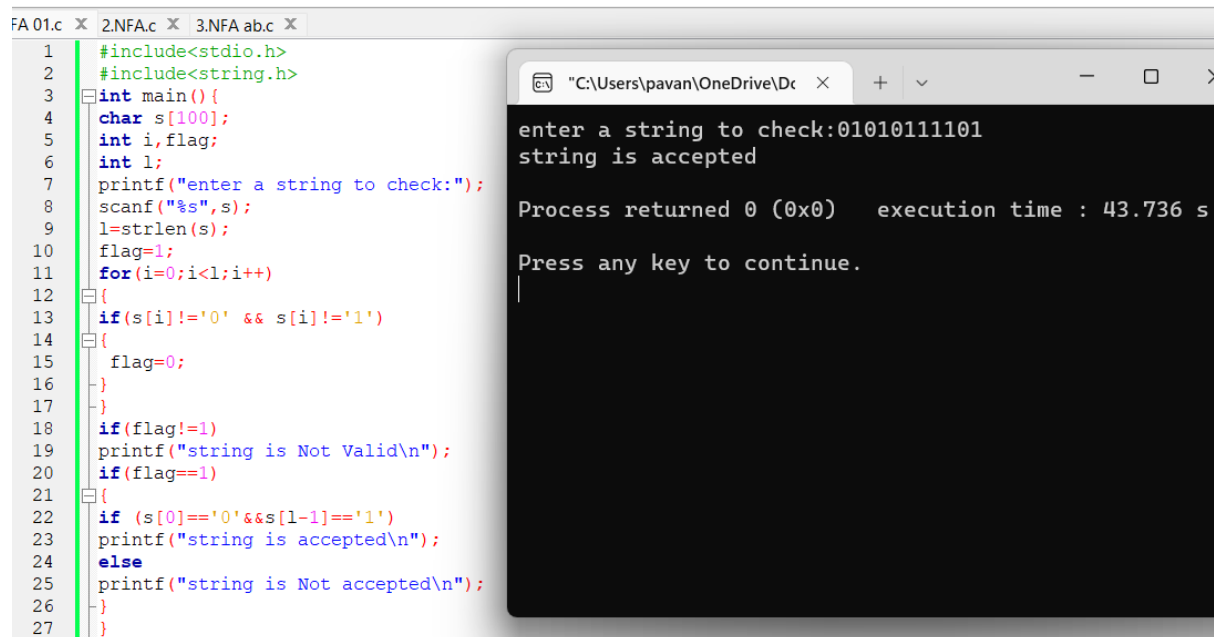
```
#include<stdio.h>

#include<string.h>

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");
}
```

```
}
```

### Output :-



The screenshot shows a C program in a text editor and its execution in a terminal window. The program checks if a string is valid based on the grammar  $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \epsilon$ . The input string is "01010111101", and the output is "string is accepted".

```
1 #include<stdio.h>
2 #include<string.h>
3 int main(){
4     char s[100];
5     int i,flag;
6     int l;
7     printf("enter a string to check:");
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        {
15            flag=0;
16        }
17    }
18    if(flag!=1)
19        printf("string is Not Valid\n");
20    if(flag==1)
21    {
22        if (s[0]=='0'&&s[l-1]=='1')
23            printf("string is accepted\n");
24        else
25            printf("string is Not accepted\n");
26    }
27 }
```

enter a string to check:01010111101  
string is accepted  
Process returned 0 (0x0) execution time : 43.736 s  
Press any key to continue.  
|

**5. To write a C program to check whether a string belongs to the grammar  $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \epsilon$ .**

### Program :-

```
#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++)

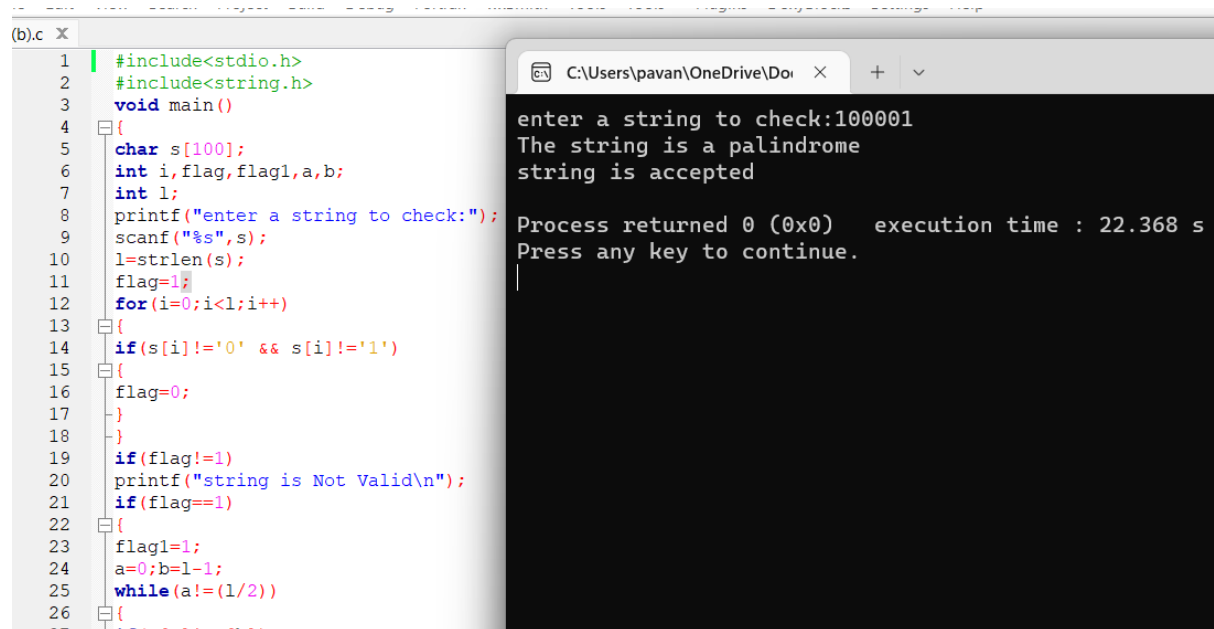
{

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");
}
else
{
printf("The string is not a palindrome\n");
printf("string is Not accepted\n");
}
```

```
}  
  
}
```

### Output :-



The screenshot shows a C program in a text editor and its execution output in a terminal window. The program checks if a string is a palindrome. The user enters '100001', and the program outputs 'The string is a palindrome' and 'string is accepted'. The terminal also shows 'Process returned 0 (0x0) execution time : 22.368 s' and 'Press any key to continue.'.

```
(b).c X  
1 #include<stdio.h>  
2 #include<string.h>  
3 void main()  
4 {  
5     char s[100];  
6     int i,flag,flag1,a,b;  
7     int l;  
8     printf("enter a string to check:");  
9     scanf("%s",s);  
10    l=strlen(s);  
11    flag=1;  
12    for(i=0;i<l;i++)  
13    {  
14        if(s[i]!='0' && s[i]!='1')  
15        {  
16            flag=0;  
17        }  
18    }  
19    if(flag!=1)  
20        printf("string is Not Valid\n");  
21    if(flag==1)  
22    {  
23        flag1=1;  
24        a=0;b=l-1;  
25        while(a!=(l/2))  
26        {  
27            if(s[a]!=s[b])  
28            {  
29                flag1=0;  
30            }  
31            a++;b--;  
32        }  
33    }  
34    if(flag1==1)  
35        printf("The string is a palindrome\n");  
36    else  
37        printf("string is not a palindrome\n");  
38 }
```

enter a string to check:100001  
The string is a palindrome  
string is accepted  
Process returned 0 (0x0) execution time : 22.368 s  
Press any key to continue.

**6. To write a C program to check whether a string belongs to the grammar  $S \rightarrow 0S0 \mid A$   
 $A \rightarrow 1A \mid \epsilon$ .**

### Program :-

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

}

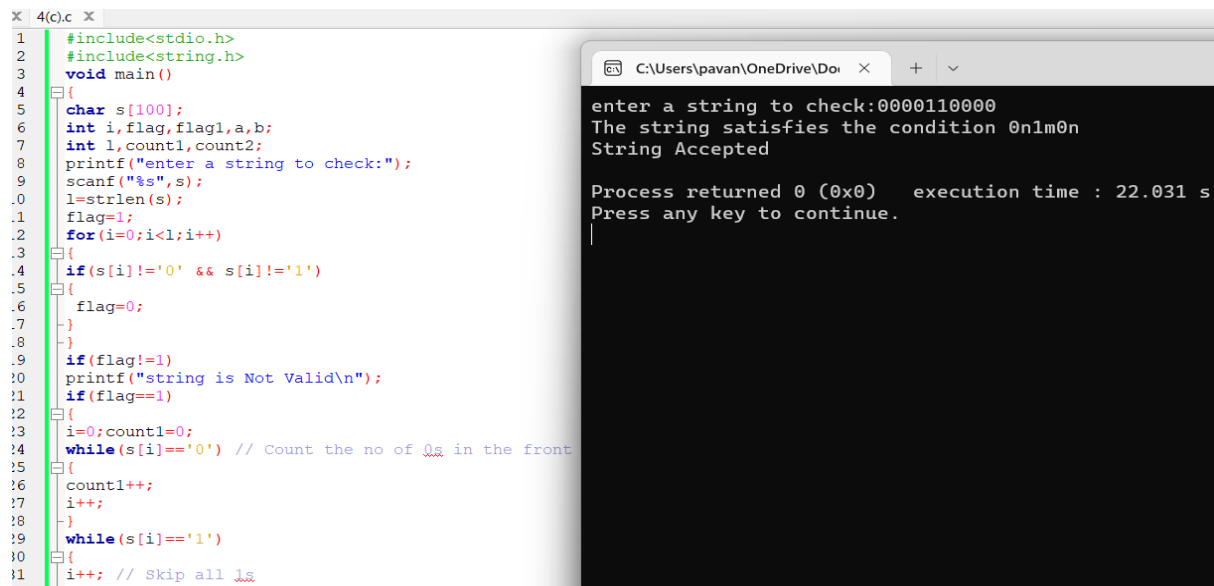
i++;

}

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");
}
}
}
```

**Output :-**



The image shows a code editor window on the left and a terminal window on the right. The code editor displays a C program that checks if a string is valid based on the grammar S -> 0 S 1 | ε. The program includes headers for stdio and string, defines a main function, and uses variables to track the string length and validity. It loops through the string, checking for '0' and '1' characters and updating a flag and count accordingly. The terminal window shows the program's execution with the input string '0000110000', the output 'The string satisfies the condition 0n1m0n', and the message 'String Accepted'. It also shows the process return code and execution time.

**7. To write a C program to check whether a string belongs to the grammar  $S \rightarrow 0 S 1 \mid \epsilon$ .**

**Program :-**

```
#include<stdio.h>

#include<string.h>

void 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;

}

}

}
```

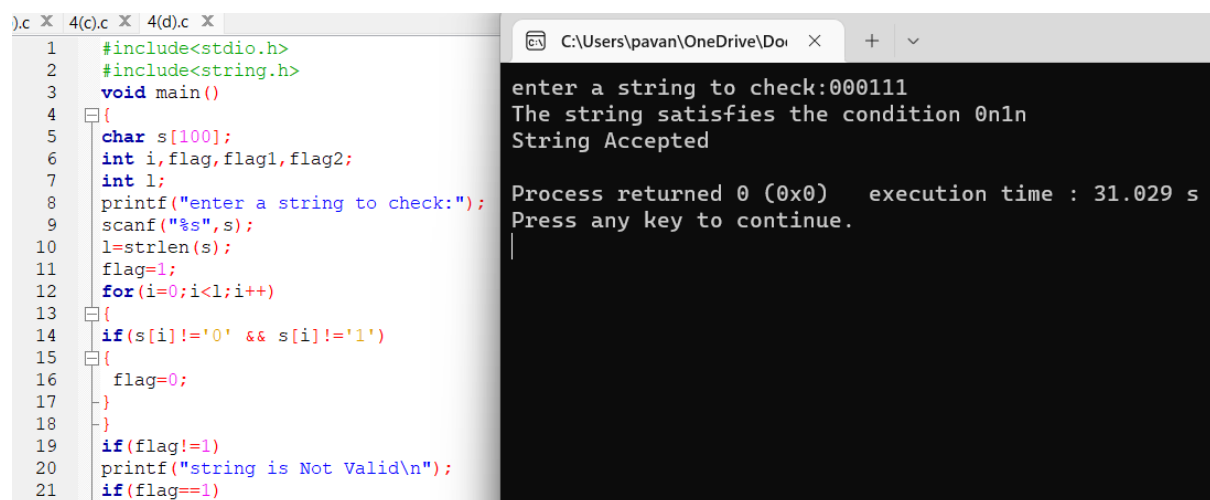
```
if(flag!=1)
printf("string is Not Valid\n");
if(flag==1)
{
if(l%2!=0) // If string length is odd
{
printf("The string does not satisfy the condition 0n1n\n");
printf("String Not Accepted\n");
}
else
{
// To check first half contains 0s
flag1=1;
for(i=0;i<(l/2);i++)
{
if(s[i]!='0')
{
flag1=0;
}
}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

```

### Output :-



The screenshot shows a C program in a code editor and its execution output in a terminal window. The code defines a function to check if a string contains only '0' and '1'. The user enters '000111', and the program outputs 'The string satisfies the condition 0n1n' and 'String Accepted'.

```

1  #include<stdio.h>
2  #include<string.h>
3  void main()
4  {
5      char s[100];
6      int i, flag, flag1, flag2;
7      int l;
8      printf("enter a string to check:");
9      scanf("%s", s);
10     l=strlen(s);
11     flag=1;
12     for(i=0;i<l;i++)
13     {
14         if(s[i]!='0' && s[i]!='1')
15         {
16             flag=0;
17         }
18     }
19     if(flag!=1)
20         printf("string is Not Valid\n");
21     if(flag==1)

```

```

enter a string to check:000111
The string satisfies the condition 0n1n
String Accepted
Process returned 0 (0x0)   execution time : 31.029 s
Press any key to continue.

```

**8 . To write a C program to check whether a string belongs to the grammar  $S \rightarrow A 1 0 1$   
 $A A \rightarrow 0 A \mid 1 A \mid \epsilon$ .**

### Program :-

```

#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;

}

}

}

if(flag1==0)

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

}

}

```

### Output :-

```

1  #include<stdio.h>
2  #include<string.h>
3  int main()
4  {
5      char s[100];
6      int i, flag, flag1;
7      int l;
8      printf("enter a string to check:");
9      scanf("%s", s);
10     l=strlen(s);
11     flag=1;
12     for(i=0; i<l; i++)
13     {
14         if(s[i]!='0' && s[i]!='1')
15         {
16             flag=0;
17         }
18     }
19     if(flag==1)
20     printf("string is Valid\n");
21     else
22     printf("string is Not Valid\n");
23     if(flag==1)
24     {

```

```

C:\Users\pavan\OneDrive\Doi  X + v
enter a string to check:000010101111
string is Valid
Substring 101 exists. String accepted

Process returned 0 (0x0)   execution time : 14.825 s
Press any key to continue.

```

**9. To write a C program to simulate a PDA for the language  $L = \{ 0^n 1^n \mid n \geq 1 \}$  in which equal number of 0's are followed by equal number of 1's.**

### Program :-

```

#include<stdio.h>

#include<string.h>

char stack[20];

int top;

void 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);
    }
}

void main()
{
    int m,i,j,k,l,a,len;
    char input[20],rem_input[20];
    printf("Simulation of Pushdown Automata for On1n\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 :-**



```
4(c).c X 4(d).c X 4(e).c X 5.Pda.c X
1  #include<stdio.h>
2  #include<string.h>
3  char stack[20];
4  int top;
5  void push()
6  {
7      top=top+1;
8      stack[top]='0';
9      stack[top+1]='\0';
10 }
11 int pop()
12 {
13     if(top<1)
14         return(0);
15     else
16     {
17         stack[top]='\0';
18         top=top-1;
19         return(1);
20     }
21 }
22 void main()
23 {
24     int m,i,k,l,j;
    ...
}
```

Simulation of Pushdown Automata for 0n1n  
Enter a string : 00001111  
Stack    Input  
Z        00001111  
Z0       0001111  
Z00      001111  
Z000     01111  
Z0000    1111  
Z000     111  
Z00      11  
Z0       1  
Z  
String accepted.....  
Process returned 13 (0xD)    execution time : 15.286 s  
Press any key to continue.