1.C program to simulate a Deterministic Finite Automata (DFA) for the given language.

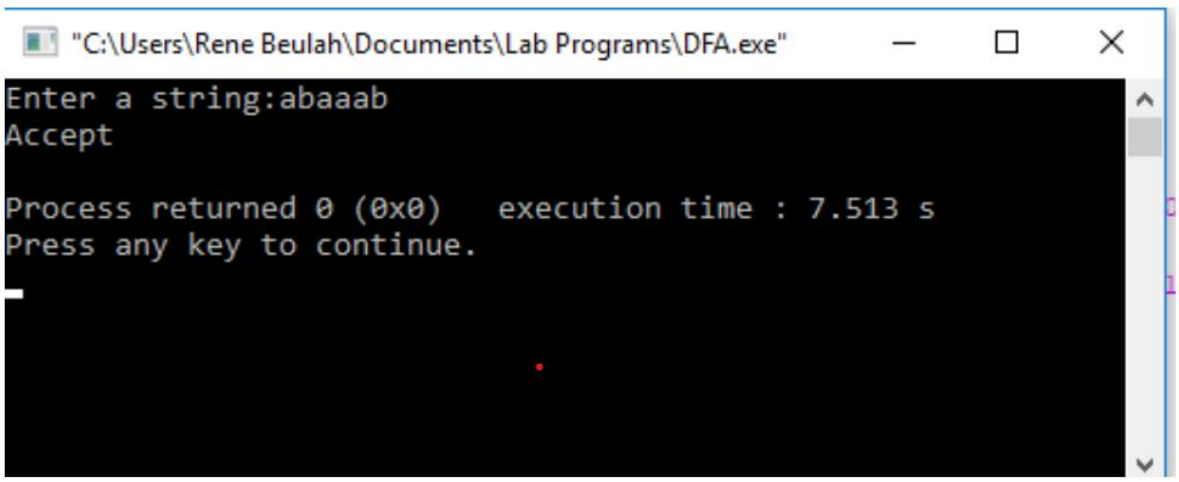
```
#include<stdio.h>
#include<string.h>
char stack[20];
int top;
void push()
{
 top=top+1;
 stack[top]='0';
 stack[top+1]='\0';
}
int pop()
{
 if(top<1)
 return(0);
 else
 {
 stack[top]='\0';
 top=top-1;
 return(1);
 }
}
void main()
{
```

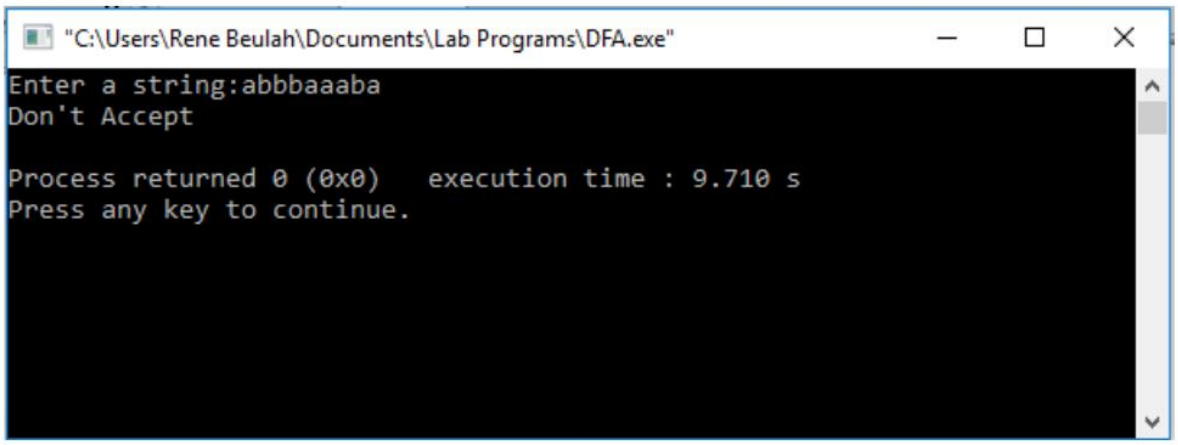
```
int m,i,j,k,l,a,len;
char input[20],rem_input[20];
printf("Simulation of Pushdown Automata for 0n1n\n");
printf("Enter a string : ");
scanf("%s",input);
l=strlen(input);
j=0;stack[0]='Z';top=0;
printf("Stack\tInput\n");
printf("%s\t%s\n",stack,input);
while(1)
{
len=strlen(input);
while(len>0)
{
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:





2. C program to simulate a Non-Deterministic Finite Automata.

```
#include<stdio.h>
#include<string.h>
char stack[20];
int top,count=0;
void push()
{
 top=top+1;
 stack[top]='0';
 stack[top+1]='\0';
}
```

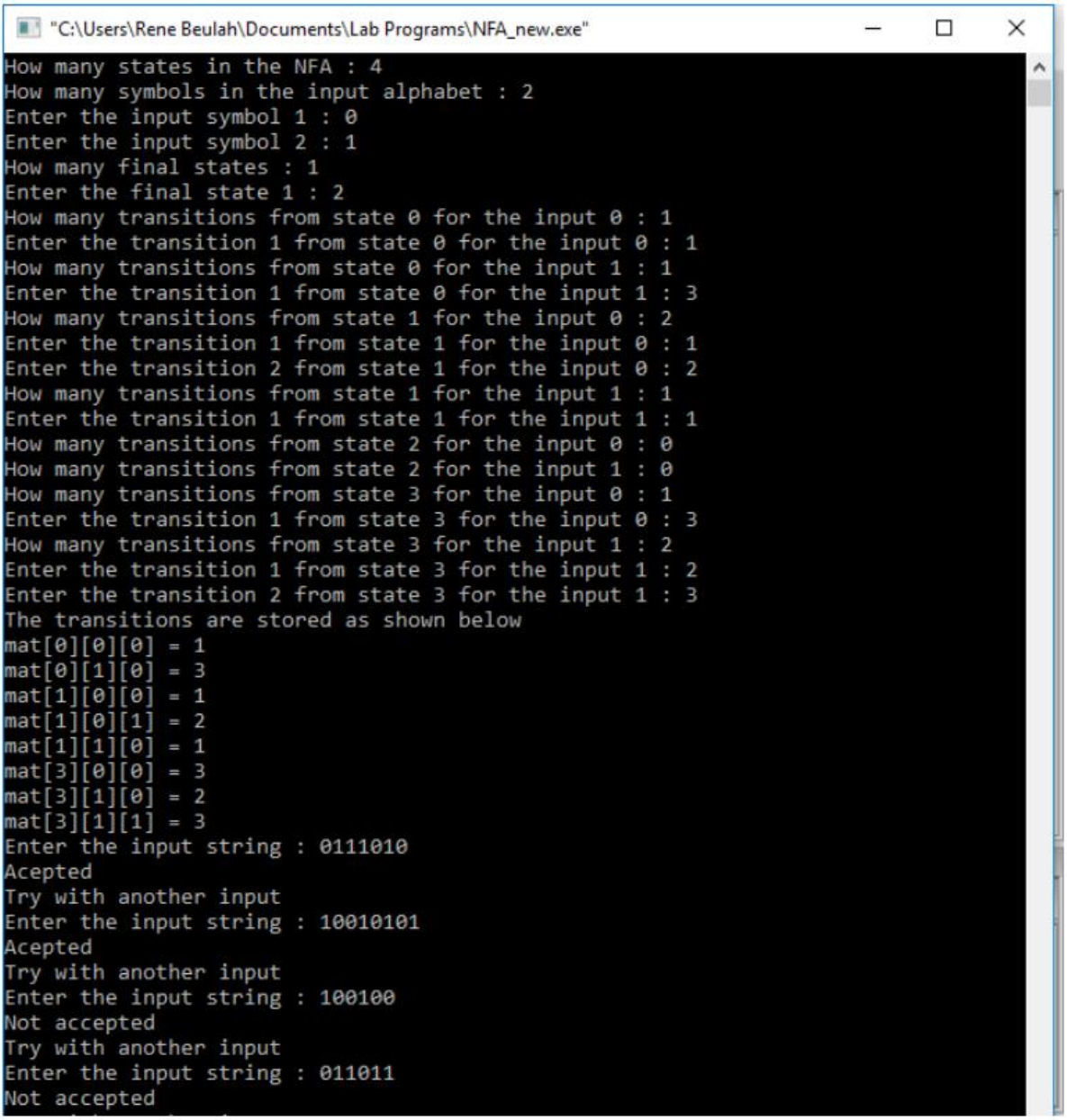
```
int pop()
{
 if(top<1)
 return(0);
 else
 {
 stack[top]='\0';
 top=top-1;
 return(1);
 }
}
void main()
{
 int m,i,j,k,l,a,len;
 char input[20],rem_input[20];
 printf("Simulation of PDA for n 0's followed by 2n 1's\n");
 printf("Enter a string : ");
 scanf("%s",input);
 l=strlen(input);
 j=0;stack[0]='Z';top=0;
 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;
//printf("j = %d\t l = %d\n",j,l);
```

```
if(j==l)
{
break;
}
}
if(top>=1)
{
printf("String not accepted");
}
else
{
printf("String accepted");
}
b:
printf(".....");
}
```

OUTPUT:



3. C program to find ε-closure of a Non-Deterministic Finite Automata with ε-moves

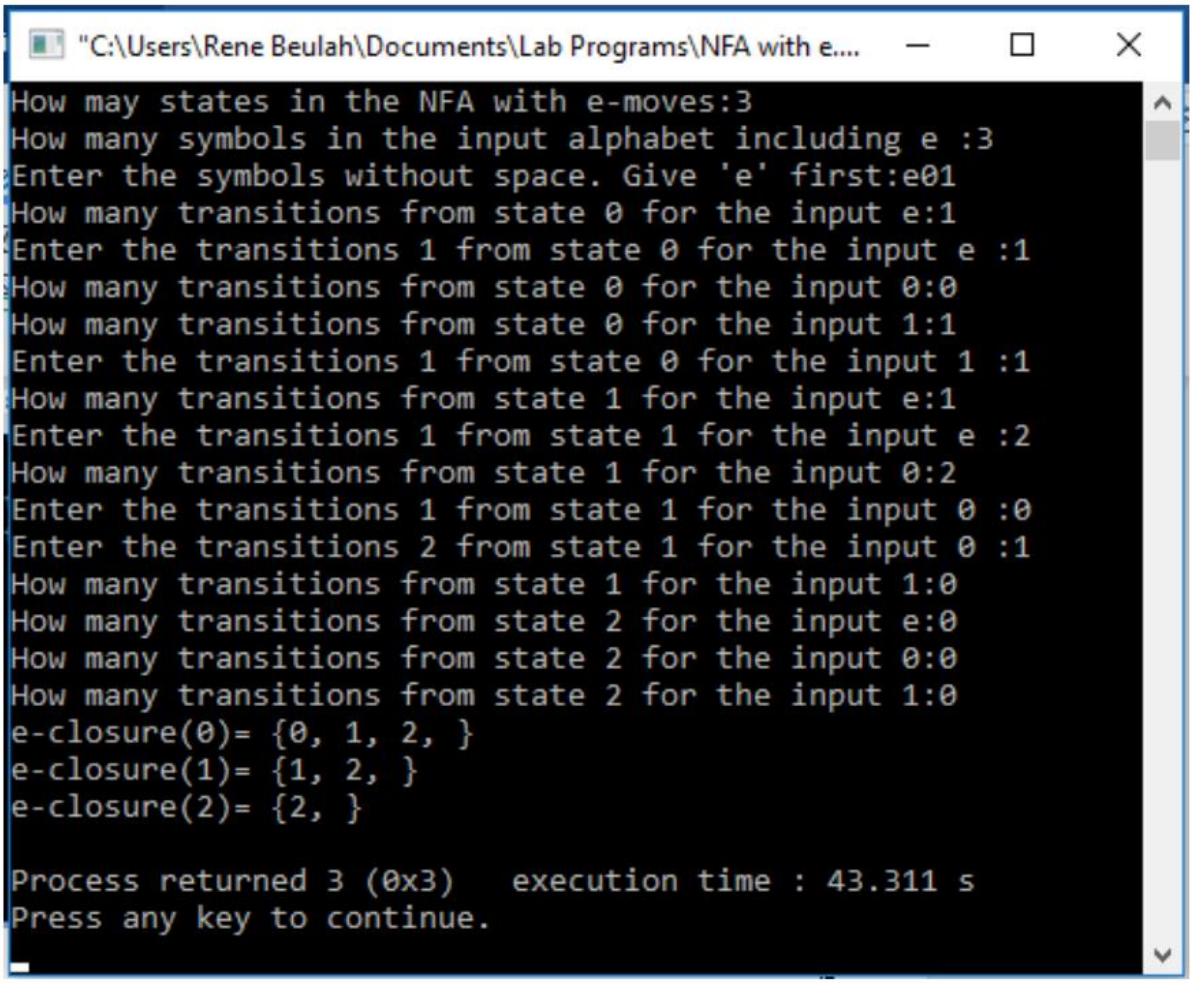
```
#include<stdio.h>
#include<string.h>
void main()
{
int i,j,le,flag,flag1,flag2;
```

```
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; //To mark that a 0 is changed to A
i=i+1;
}
else if((str[i]=='0')&&(flag==1))
{
i=i+1; //Skip 0
}
else if(str[i]=='A')
{
i=i+1; //Skip A
}
else if((str[i]=='1')&&(flag1==0))
```

```
{
str[i] = 'B';
printf("%s\n",str);
flag1=1; //To mark that a 1 is changed to B
i=i+1;
}
else if((str[i]=='1')&&(flag1==1))
{
i=i+1; //Skip 1
}
else if(str[i]=='B')
{
i=i+1; //Skip B
}
else if((str[i]=='2')&&(flag2==0))
{
str[i] ='C';
printf("%s\n",str);
flag2=1; //To mark that a 2 is changed to C
i=i+1;
}
else if((str[i]=='2')&&(flag2==1))
{
i=i+1; //Skip 2
}
else if(str[i]=='C')
{
```

```
i=i+1; //Skip C
}
}
j=j+1;
if(j==le)
{
 break;
}
}
}
```

OUTPUT:



4. C program to check whether a string belongs to the grammar

```
#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]);
```

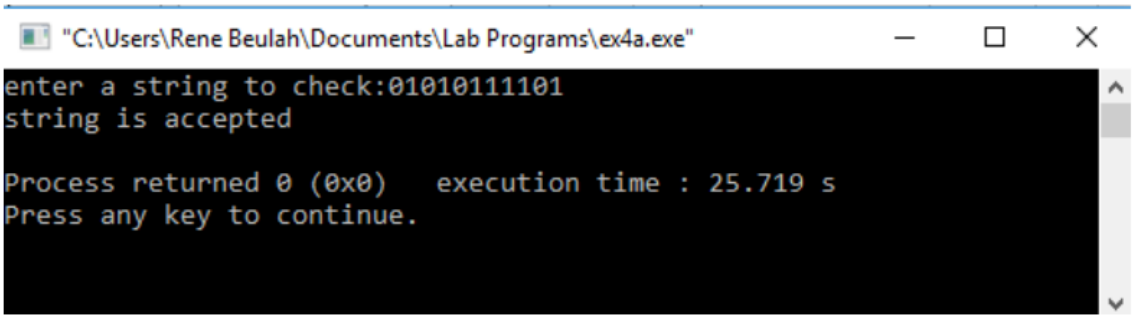
```
}
//Initialize all entries with -1 in Transition table
for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
for(k=0;k<10;k++)
{
mat[i][j][k]=-1;
}
}
}
//Get input from the user and fill the 3D transition table
for(i=0;i<num_states;i++)
{
for(j=0;j<num_symbols;j++)
{
printf("How many transitions from state %d 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]);
}
}
```

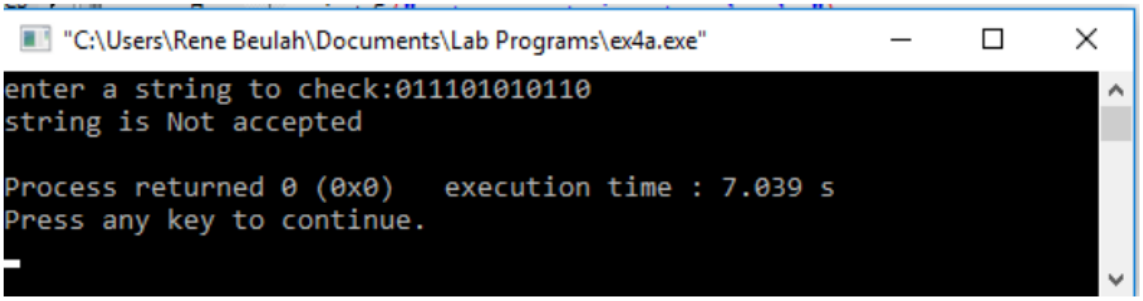
```
}
printf("The transitions are stored as shown below\n");
 for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
for(k=0;k<10;k++)
{
 if(mat[i][j][k]!=-1)
 printf("mat[%d][%d][%d] = %d\n",i,j,k,mat[i][j][k]);
}
}
}
while(1)
{
printf("Enter the input string : ");
scanf("%s",input);
present_state[0]=0;
prev_trans=1;
l=strlen(input);
for(i=0;i<l;i++)
{
 if(input[i]=='0')
 inp1=0;
 else if(input[i]=='1')
 inp1=1;
 else
```

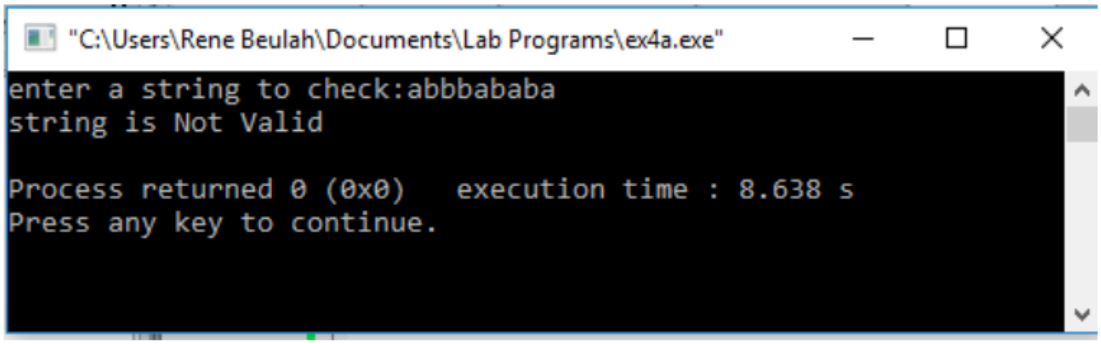
```
 {
 printf("Invalid input\n");
 exit(0);
 }
 for(m=0;m<num_symbols;m++)
{
if(inp1==symbol[m])
{
inp=m;
break;
}
}
new_trans=0;
for(j=0;j<prev_trans;j++)
{
k=0;
p=present_state[j];
while(mat[p][inp][k]!=-1)
{
next_state[new_trans++]=mat[p][inp][k];
k++;
}
}
for(j=0;j<new_trans;j++)
{
present_state[j]=next_state[j];
}
```

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

OUTPUT:





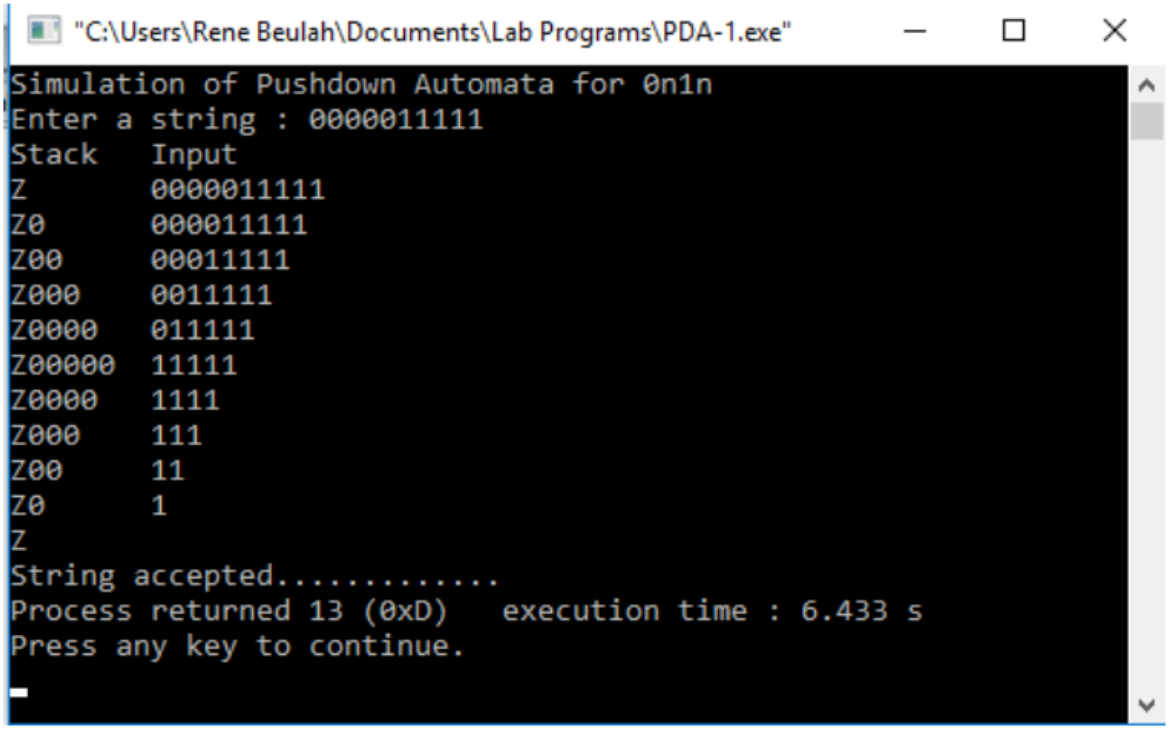


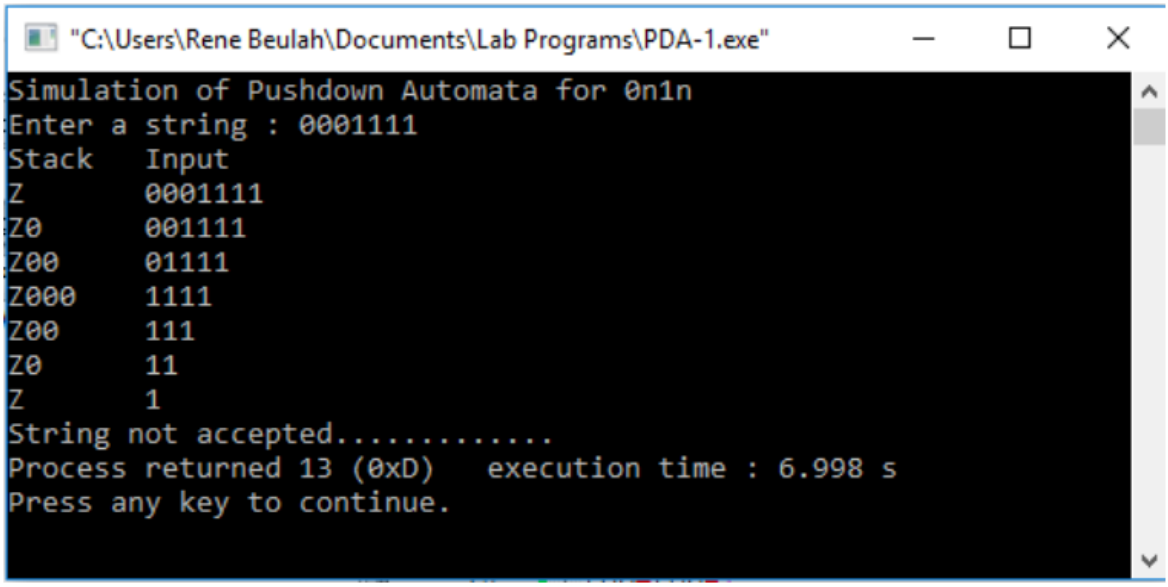
5. SIMULATING PUSHDOWN AUTOMATA(PDA)

```
#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=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");
}
```

OUTPUT:


"C:\Users\Rene Beulah\Documents\Lab Programs\PDA-1.exe"
Simulation of Pushdown Automata for 0n1n
Enter a string : 0000011111
Stack Input
Z 0000011111
Z0 000011111
Z00 00011111
Z000 0011111
Z0000 011111
Z00000 11111
Z0000 1111
Z000 111
Z00 11
Z0 1
Z
String accepted.....
Process returned 13 (0xD) execution time : 6.433 s
Press any key to continue.


"C:\Users\Rene Beulah\Documents\Lab Programs\PDA-1.exe"
Simulation of Pushdown Automata for 0n1n
Enter a string : 0001111
Stack Input
Z 0001111
Z0 001111
Z00 01111
Z000 1111
Z00 111
Z0 11
Z 1
String not accepted.....
Process returned 13 (0xD) execution time : 6.998 s
Press any key to continue.