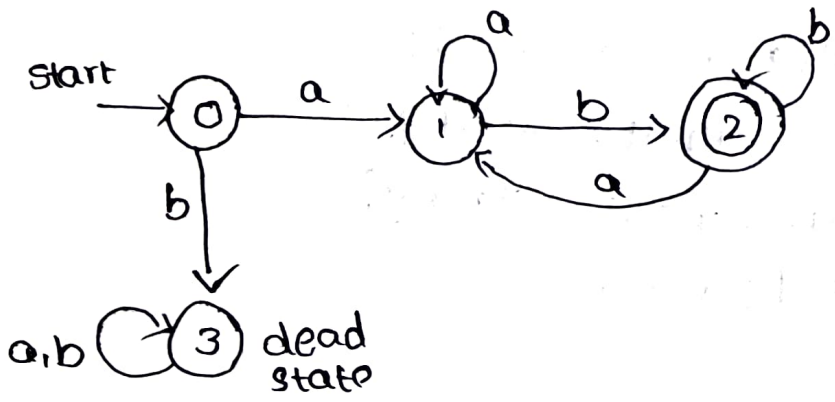
Write a C Program to Simulate a Deterministic finite Automata.

Design of the DFA



Transition Table :

State / Input	a	b
->0	1	3
1	1	2
(2)	1	2
3	3	3

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 ("Accept \n");
else
```

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

output

```
Enter a string : abaaab
Accept

Enter a string : abbbaaaba
Don't Accept
```

checking whether a string Belongs to a Grammar

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 vaiid\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 :

enter a string to check : 01010111101
string is accepted

enter a string to check : 011101010110
string is not accepted

enter a string to check : abbbababa
String is not valid.

write a c program to find ε- closure of a non- deterministic finite Automata with ε- moves.

Program

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

e-closure (0) = 0, 1

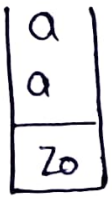
e-closure (1) = 1

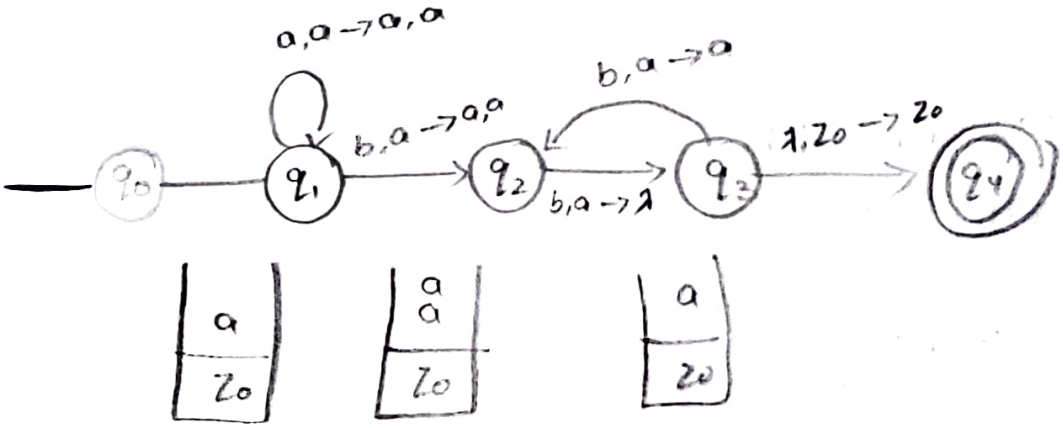
e-closure (2) = 2

Stimulators

Design PDA for input stimulator to accept the string $a^n b^{2n}$

a a b b b b





$\delta(q_0, q_1, Z_0) = (q_1, a\,Z_0)$

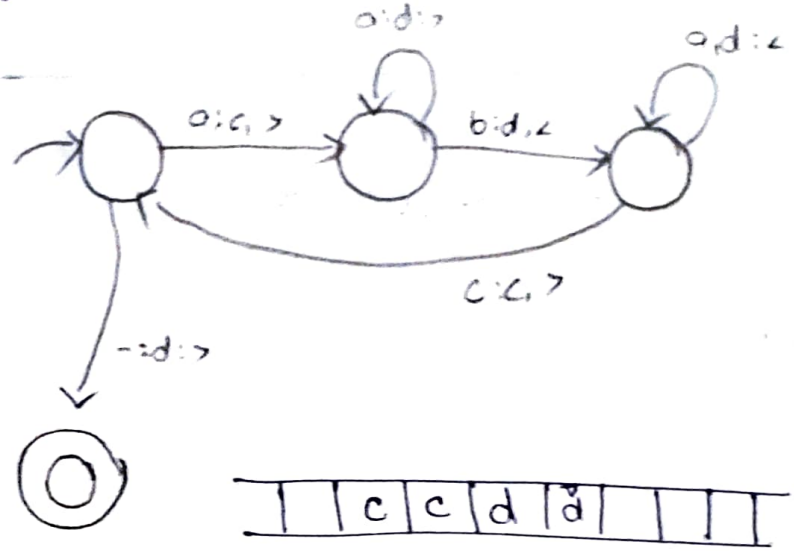
$\delta(q_1, a, a) = (q_1, aa)$

$\delta(a, b, a) = (q_2, aa)$

$\delta(q_2, b, a) = (q_3, \lambda)$

$\delta(q_3, \lambda, Z_0) = (q_4, Z_0)$

Design TM to accept the input storing $A^n B^n$



Design TM to accept the input storing $A^n B^{2n}$

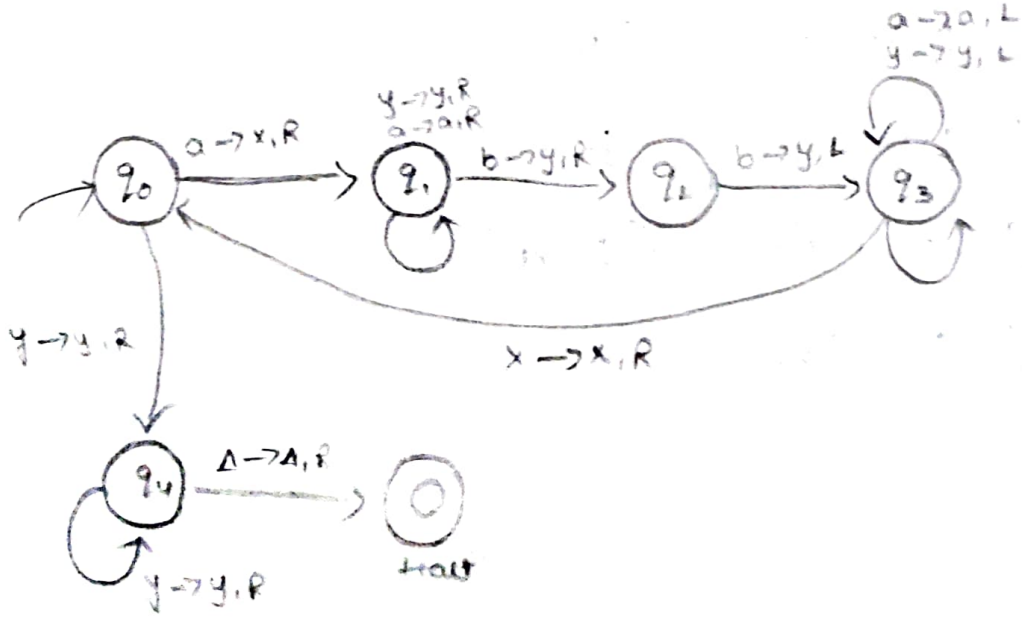
aa bb bb

x a Y Y

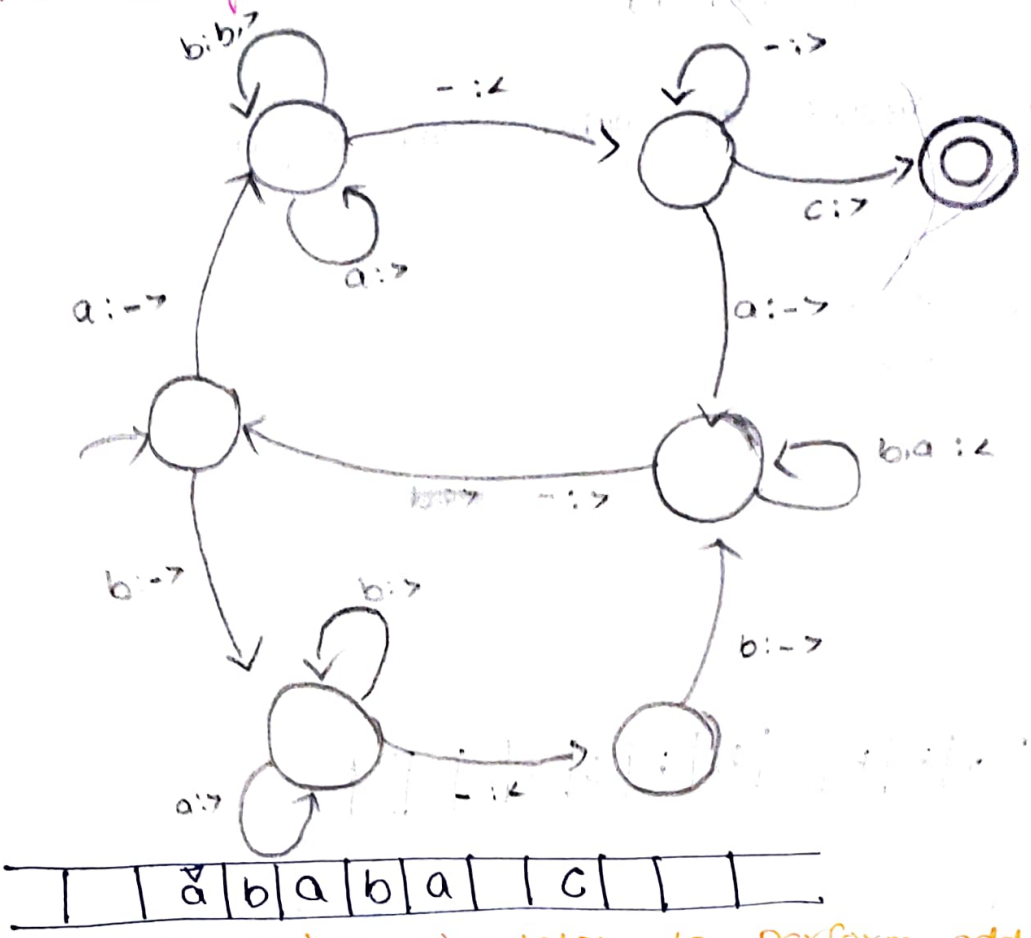
x Y Y Y Y

x Y Y Y Y

→ halt



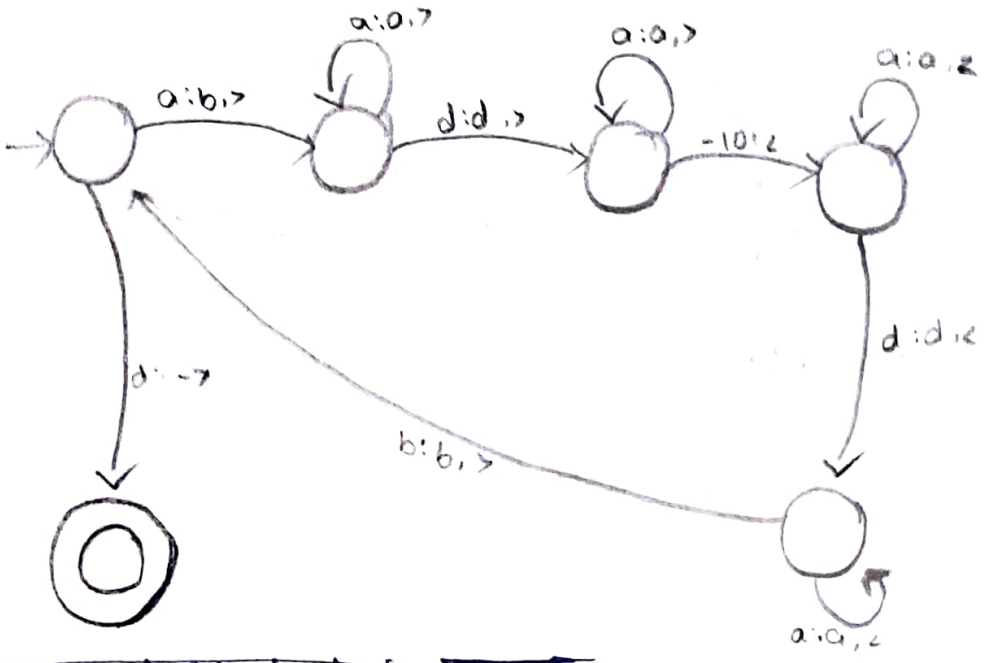
Design TM using simulator to accept the input string Palindrome



		ä	b	a	b	a		c		

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

$\omega = aa + aaaa$

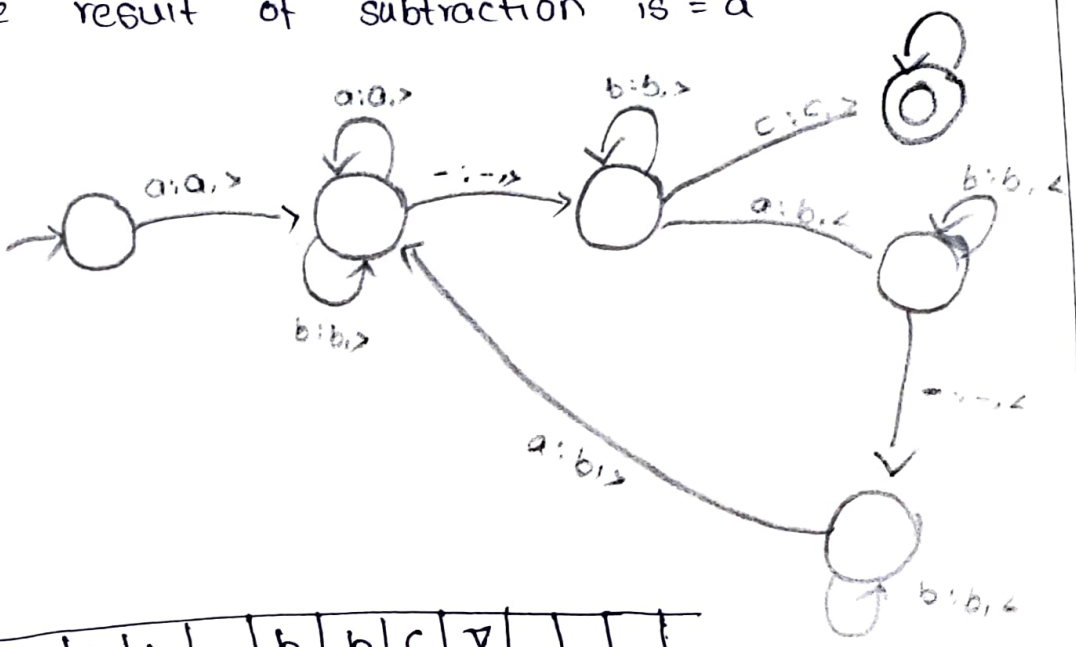


	b	b		ä	a	a	a	a	a	

Design TM to Perform subtraction.

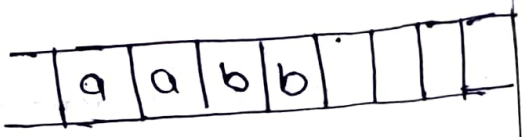
$w = aaa - aa$

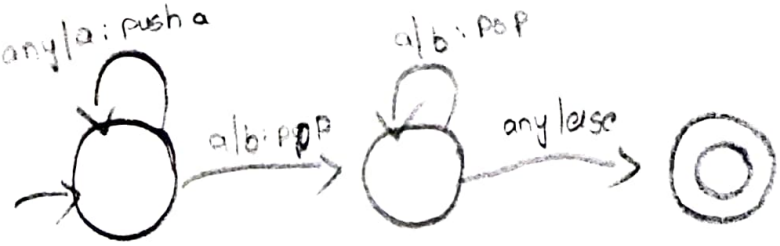
The result of subtraction is = a



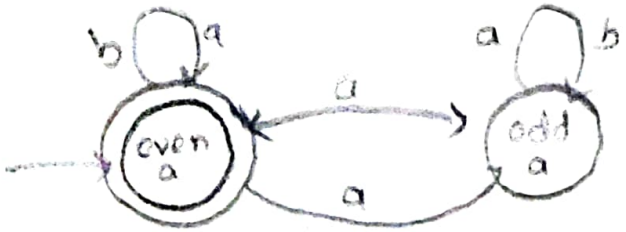
Design PDA using simulator to accept the input string aabb.

$a^n b^n = aabb$

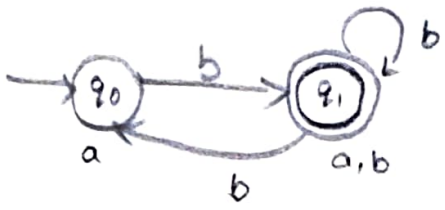




Design DFA to accept even number of a's

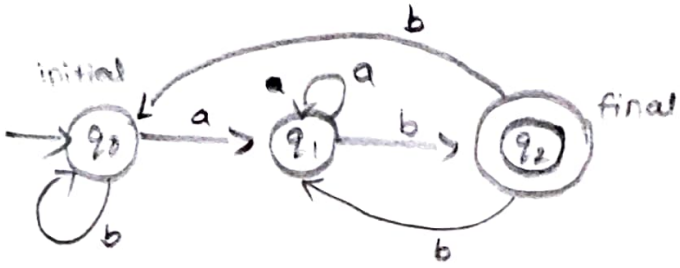


Design DFA to accept odd number of a's

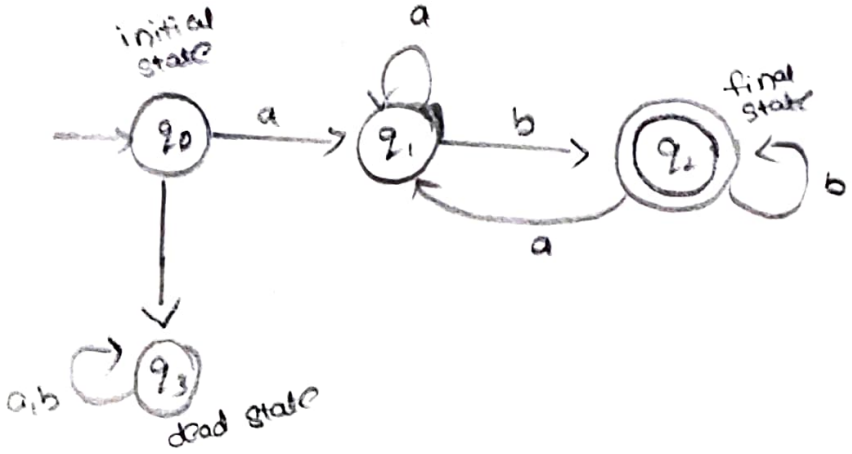


Design DFA to accept the string that with ab over {a,b}

ω = aaabab



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



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

$\Sigma = \{a,b\}$