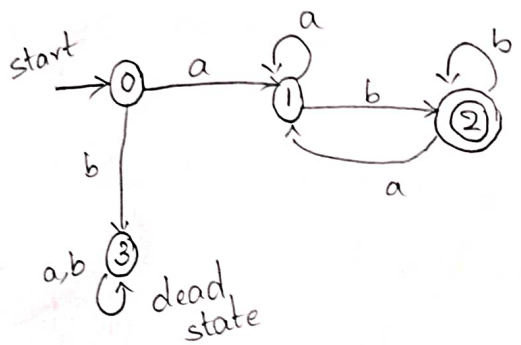
① write a C program to simulate a DFA for the given language representing strings that start with a and end with b

Aim :

To create a C program to simulate a DFA for the given language representing strings that start with a end with b.

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 transtable[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", inputstring);
fnl = strlen(inputstring);
for (i=0; i<fnl; i++)
{
if (inputstring[i] == 'a')
next_state = trans_table[present_state][0];
else if (inputstring[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

Result :-

—Hence To create a C program to simulate a DFA for the given language representing string that start and end with b executed successfully

② Checking whether a string belongs to a grammar

Aim :-

To create a C program to check whether a string belongs to a grammar

Algorithm:

1. Get the input string from the user
2. find the length of the string
3. Check whether all the symbols in the input are either 0 or 1. If so.
4. print "string is valid" and so the step 5: other wise print "string not valid" and quit the program.
6. If the first symbol is 0 and the last symbol is 1 print "string accept" otherwise print "string not accepted"

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=1; i<l; i++)
```

```
{
  if [s[i] :'0' && g[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 :

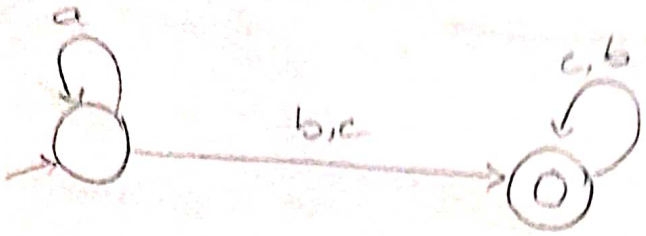
Enter a string to check : 01010111101

string is accepted.

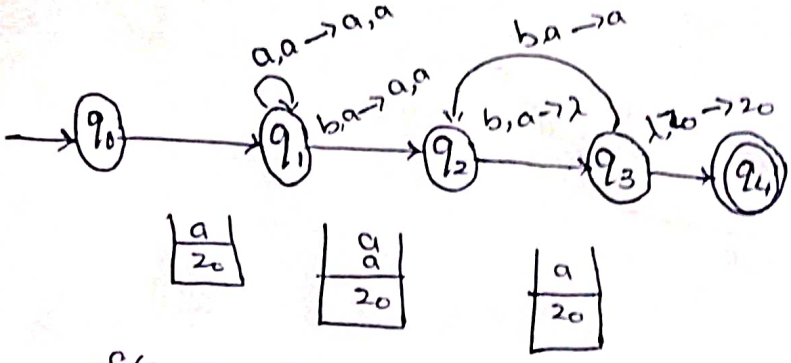
Result :-

Hence to check whether a string brings to a grammar is Gnerated successfully.

(12) Design DFA using simulator to accept the input string "a", "ac" and "bac".



(13) Design PDA using simulator to accept the input string aabb



$\delta(q_0, a, Z_0) = (q_1, aZ_0)$

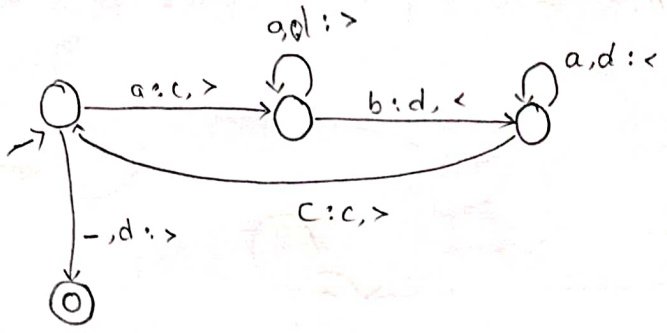
$\delta(q_1, a, a) = (q_1, aa)$

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

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

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

(15) Design TM to accept the input string $A^n B^n$



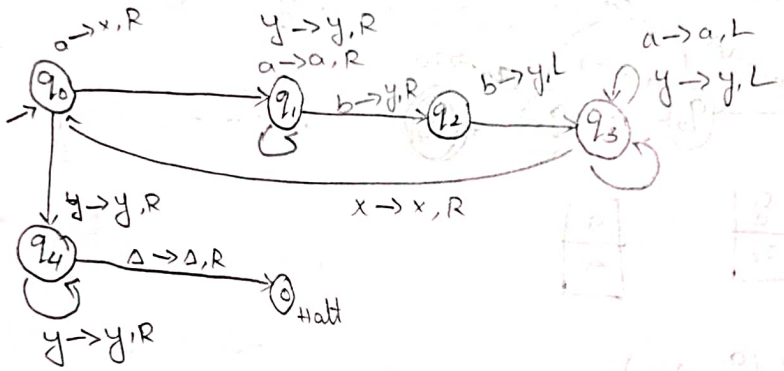
(16) Design TM to accept the input string $A^n B^{2n}$

aa bb bb

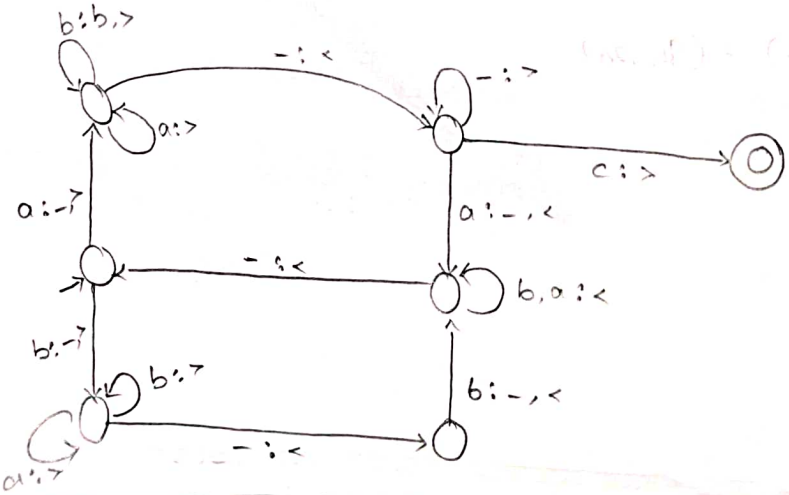
xa yy ←

x Y Y Y Y ←

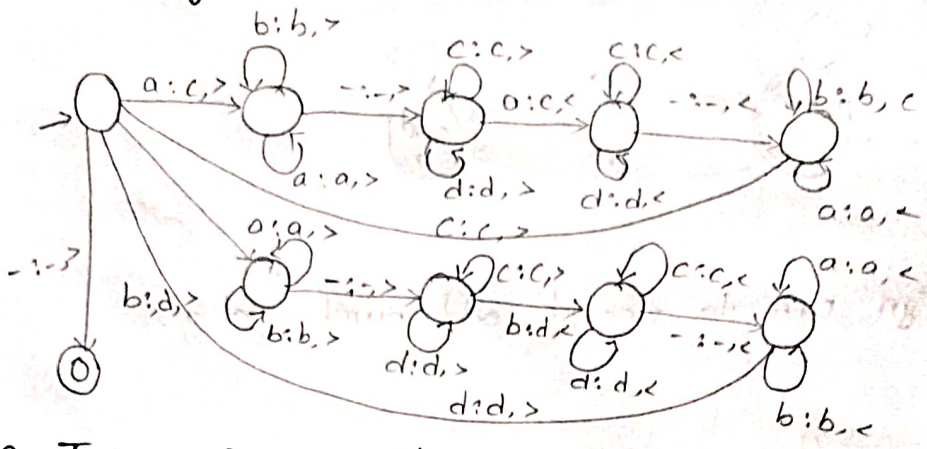
x → Y Y Y Y



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



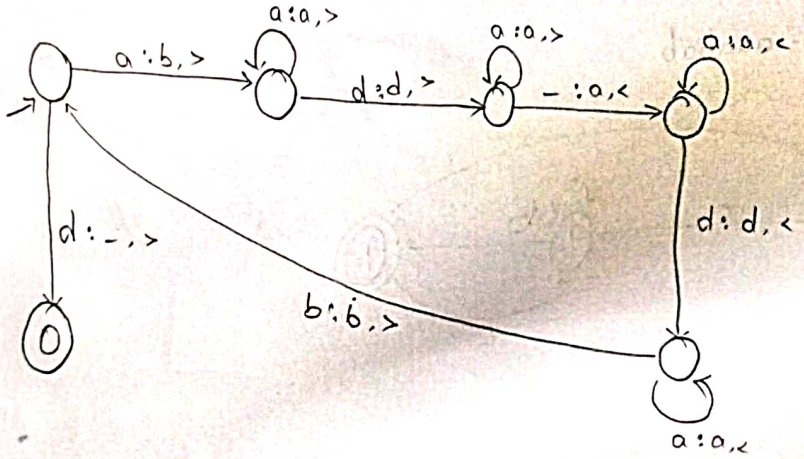
(18) Design TM using simulator to accept input string perform string comparison w = aba aba



(19) Design TM using simulator to ~~accept~~ perform addition of 'aa' and 'aaa'

W = aa + aaaa

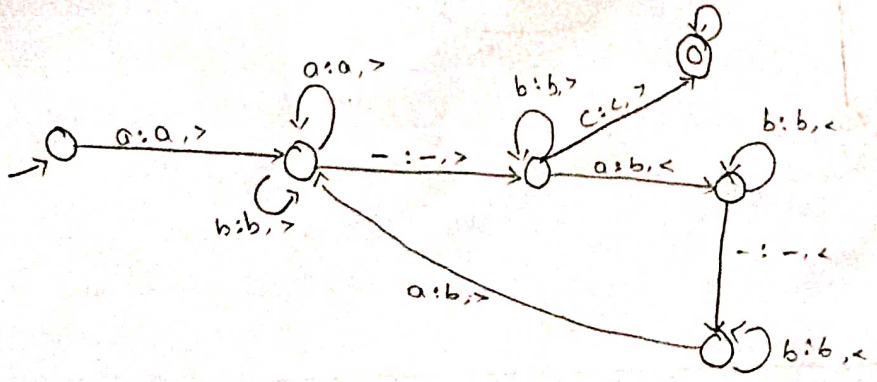
After Addition of a's = aaaaaa



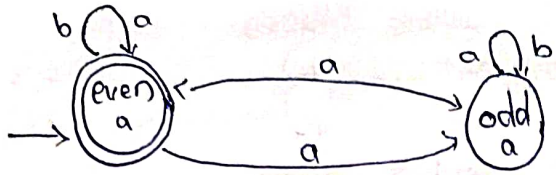
(20) Design TM using simulator to perform subtraction of 'aaa' and 'aa'

w = aaa - aa

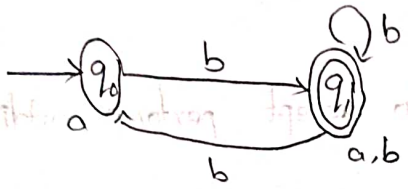
The Result of subtraction is = a



(21) Design DFA to accept even number of a's

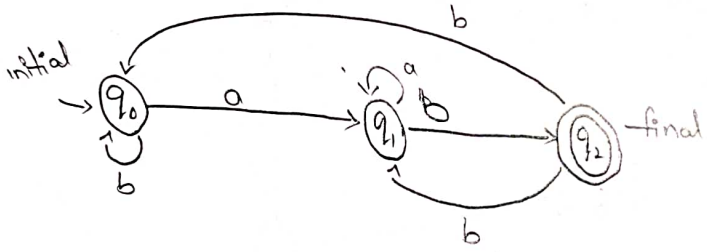


(22) Design DFA to accept odd number of a's

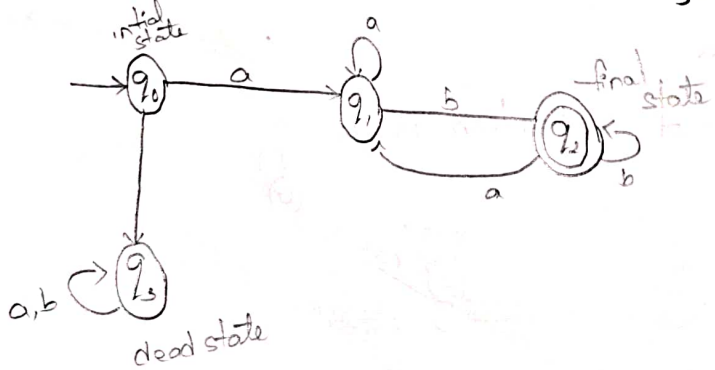


(23) Design DFA to accept the string that with ab over {a,b}

$w = aaabab$

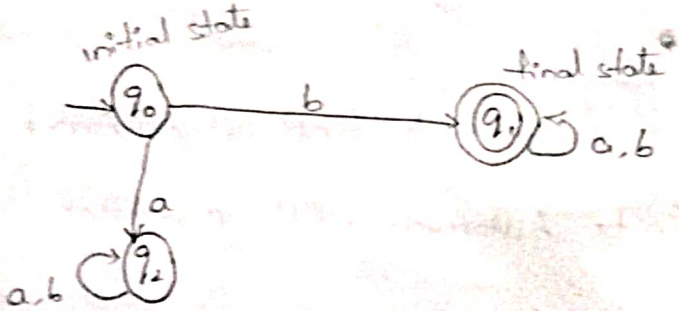


(24) Design DFA using simulator to accept the string having 'ab' as substring over the set {a,b}

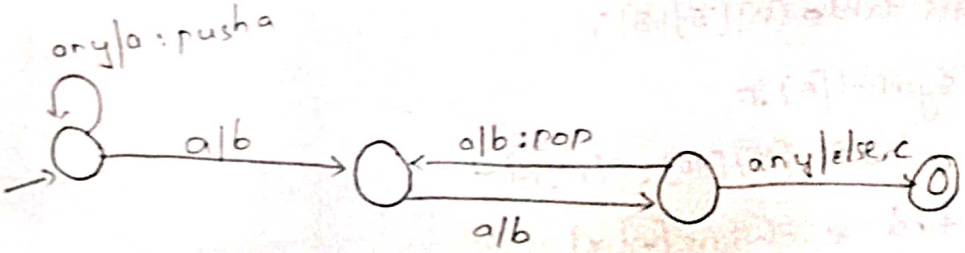


(ii) Design DFA using simulator to accept the string start with a or b over the set {a,b}

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



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



Finding ε-closure for NFA with ε-moves

AIM :

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

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, numStates, 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 may 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++)
{
  print ("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 transition %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++)
```

```
}
  for(i=0 ; i<num_states ; i++)
   e_closure[i][0]=1 ;
  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-cloure (%d) = {", i) ;
   for (i=0 ; j<num_states ; j++)
  {
    if (e_closure (i)(j) = -1 )
{
   printf ("%d ," , e_closure (i)(j)) ;
  }
    print (" }\n") ;
}
}

    void find_e_closure (int x)
  {
  int i, j, y(10) , numtrans ;
   i=0 ;
while (trans_table(x) (10)(i) ! = -1 )
{
```

```
numtrans = 1;
for (j=0; j<=numstates; j++)
{
    e_closure(table)[ptr].y[j];
    ptr++;
    find_e_closure(y[j]);
}
}
```

Output :-

e-closure (0) = {0, 1, 2, }

e-closure (1) = {1, 2, }

e-closure (2) = {2, }

Result :-

Hence the C program to find e-closure of a non-deterministic finite Automata with ε-moves. Generated Successfully.