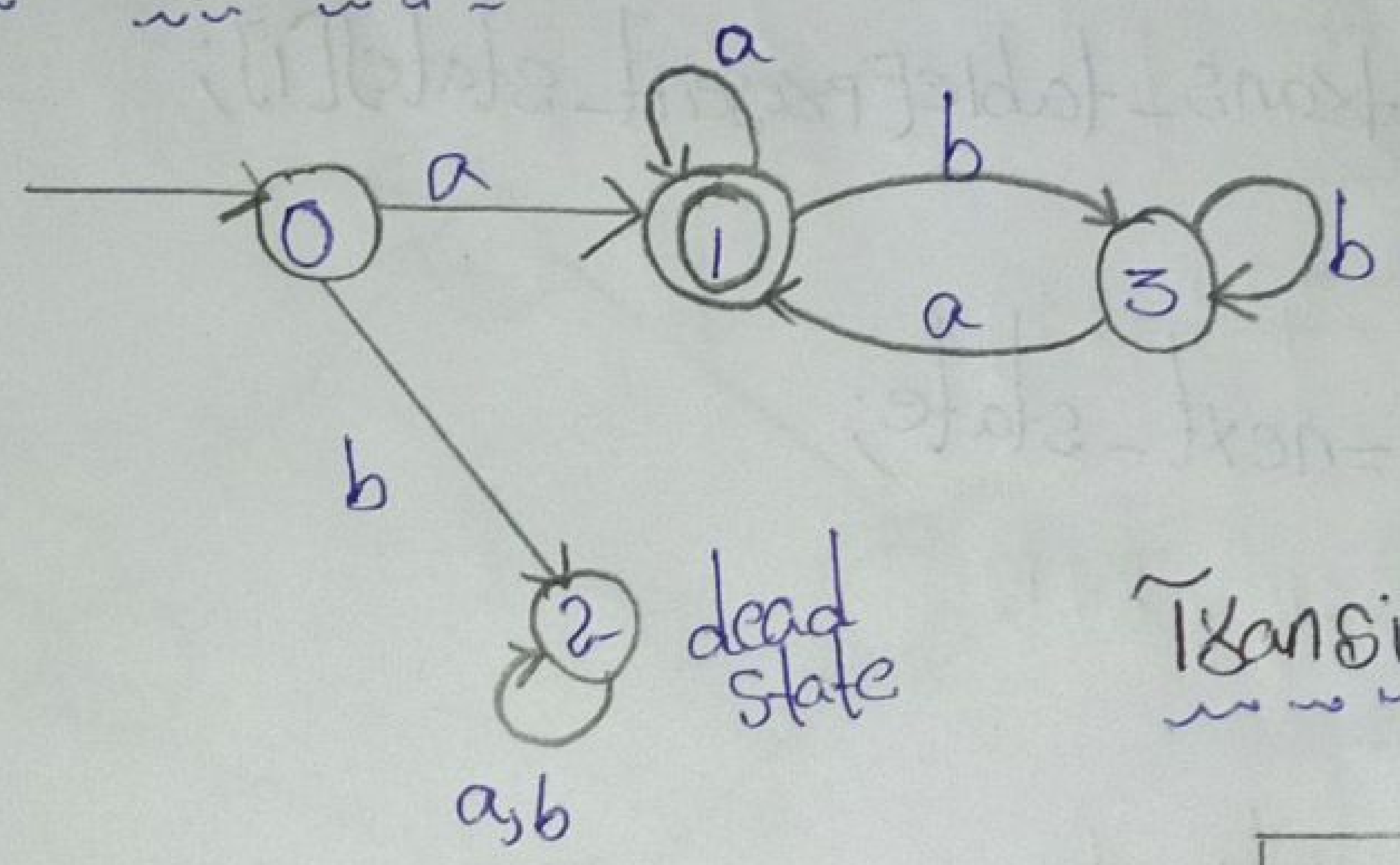
(1) Write a c Program to stimulate a Deterministic finite Automata (DFA).

AIM :~

To write a c Program to simulate a Deterministic finite Automata.

Design of DFA :~



Transition table:

Present	Next	
	a	b
→0	1	2
1	1	3
2	2	2
3	1	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;
    int 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 input string :abaabab

Accept.

Result :~

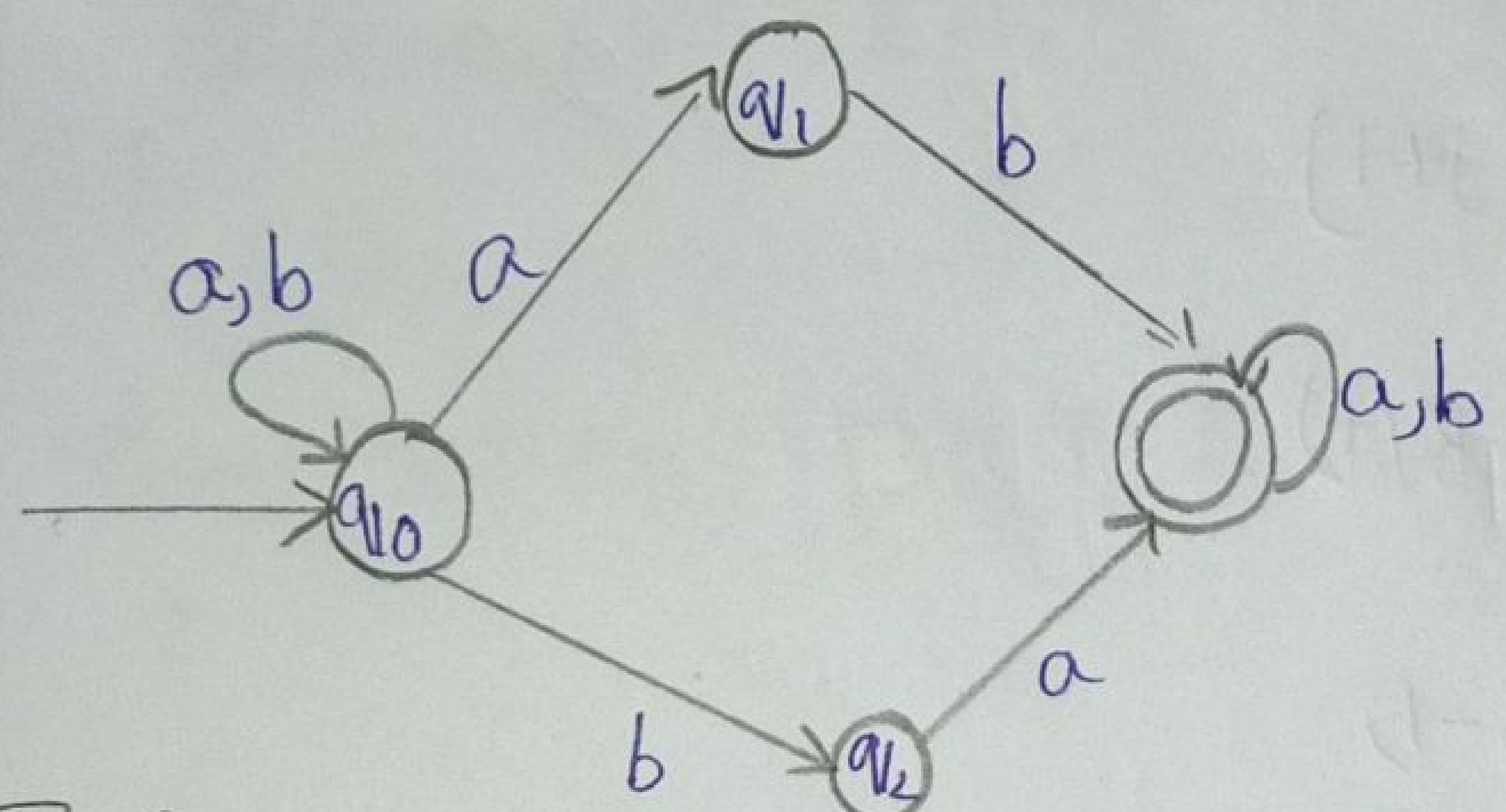
Thus the given c Program for DFA is Executed Sucessfully.

(2) Write a Program to stimulate Non Deterministic finite Automata strings that start with a and end with b.

AIM :~

To Perform of c Program for Non-Deterministic finite Automata.

Program Design :~



state/input	0	1
→0	1	3
1	{1,2}	1
②	–	–
3	3	{2,3}

Program :~

```
#include <stdio.h>
#include <string>
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_tran,new_trans;
   Char ch,input[20];
   int symbol[5],inp,inp1;
Printf("How many symbols states in the NFA:");
```

Scanned by TapScanner

```
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 states":);
}
 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(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]);
    }
  }
}
Printf("enter the input string:");
Scanf("%s",input);
Present_state[0]=0;
Prev_trans=1;
for(i=0;i<l;i++)
{
  if(input[i]=='0')
inp1=0;
  else
  {
  Printf("invalid input\n");
```

```
}
for(m=0; m<num_symbols; m++)
  {
    if(inp1 = =symbol[m])
    {
        inp =m;
        break;
    }
  }
  while(mat[p][inp][k]! =-1)
  {
   next_state[new_trans++] = mat[p][inp][k];
   k++;
  }
 }
 for(j=0; j<new_trans; j++)
 {
  flag = 0;
  for(i=0; i<prev_trans; i++)
 {
  for(j=0; j<num_final; j++)
  {
      flag = 1
      break;
  }
 }
}
 Printf("Accepted\n");
else
 Printf("Not accepted\n");
Printf("Try with another input\n");
}
}
```

Scanned by TapScanner

Output :~

How many states in the NFA : 4
How many symbols in the input alphabet : 2.
Enter the input symbol1 : 0
Enter the input symbol2 : 1
Enter the input symbol3 : 1
How many final states : 1
Enter the final state : 1
The transitions are stored as shown below

mat [0] [0] [0] = 1
mat [0] [1] [0] = 3
mat [0] [1] [1] = 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.

Result :~

Thus the given C Program is executed sucessfully.

(5) checking wheather a string belongs to a grammer.

Aim :-

To create a C Program to check wheather a string belongs to a grammer.

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' && s[i] != '1')

{
flag = 0;
```

Scanned by TapScanner

```
    }
}
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

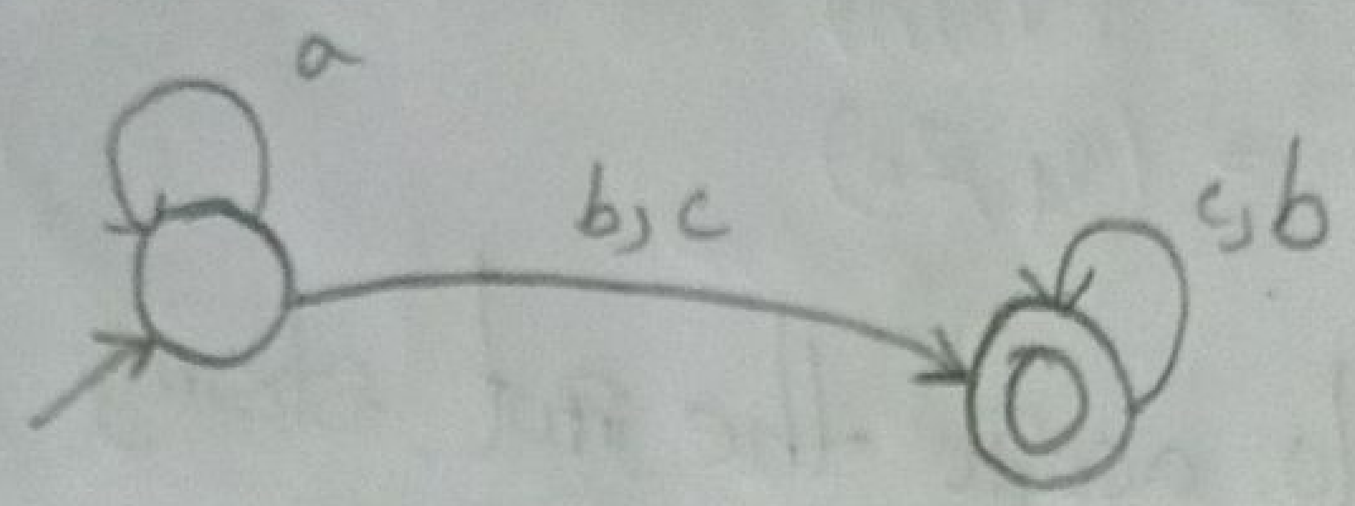
string is accepted

Result :

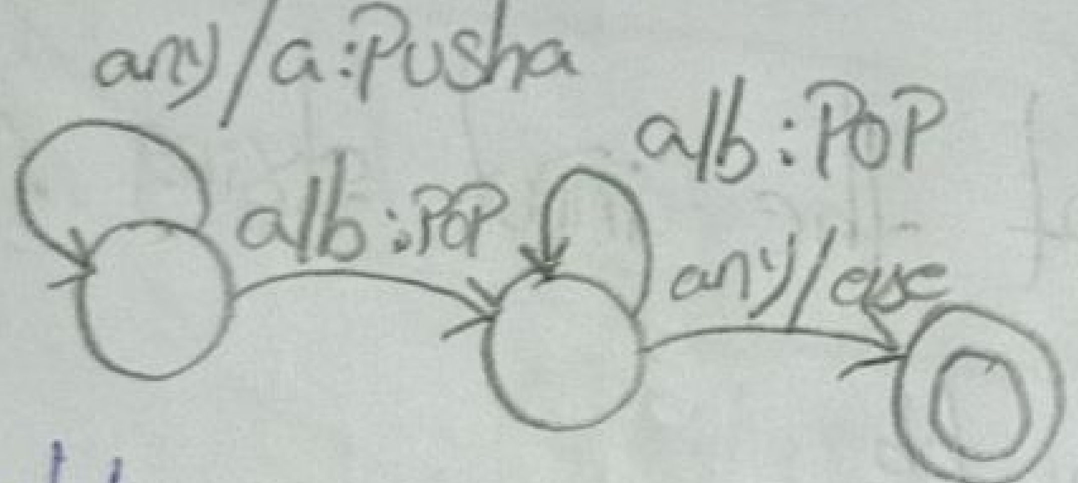
Thus the given Program to check the string belongs to a grammar or not is executed successfully.

Simulators:

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

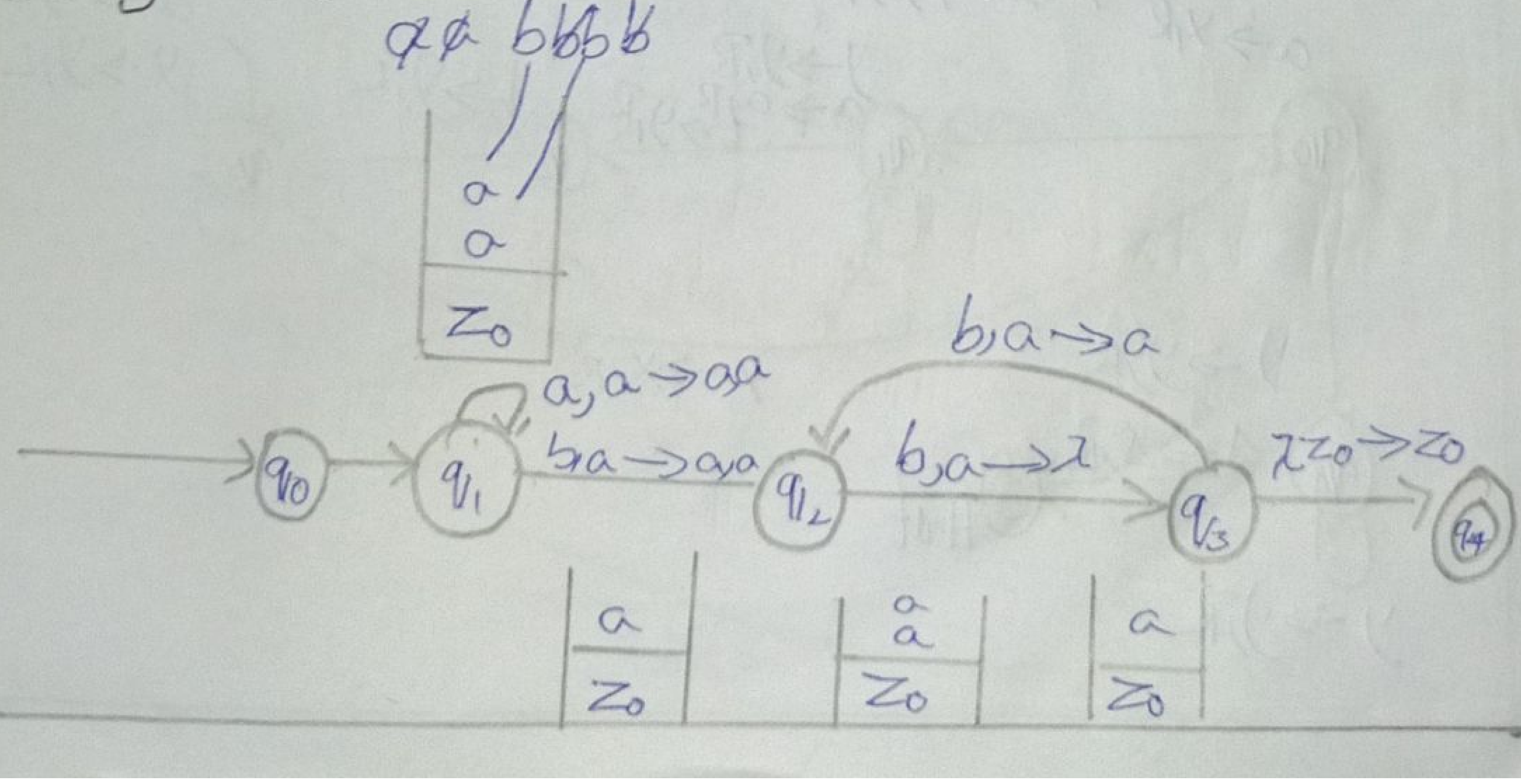


14) Design PDA using simulator to accept the input string aabb.



Input : aabb

(14) Design PDA using simulator to accept the input string $a^n b^n$



$\delta(q_0, a, z_0) = (q_1, az_0)$

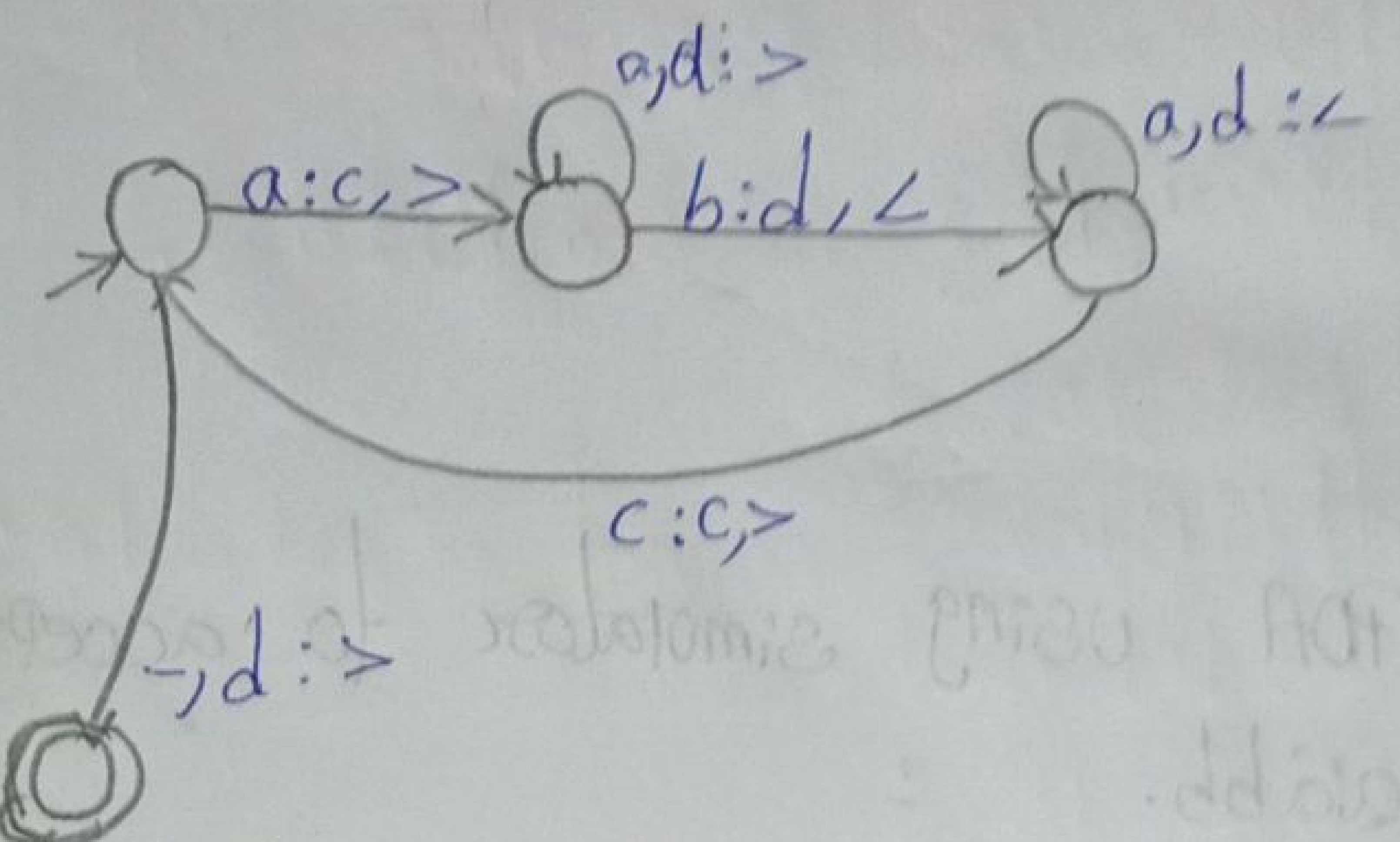
$\delta(q_1, a, a) = (q_1, aa)$

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

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

$\delta(q_3, \lambda, z_0) = (q_4, z_0)$

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



Input: aabbab

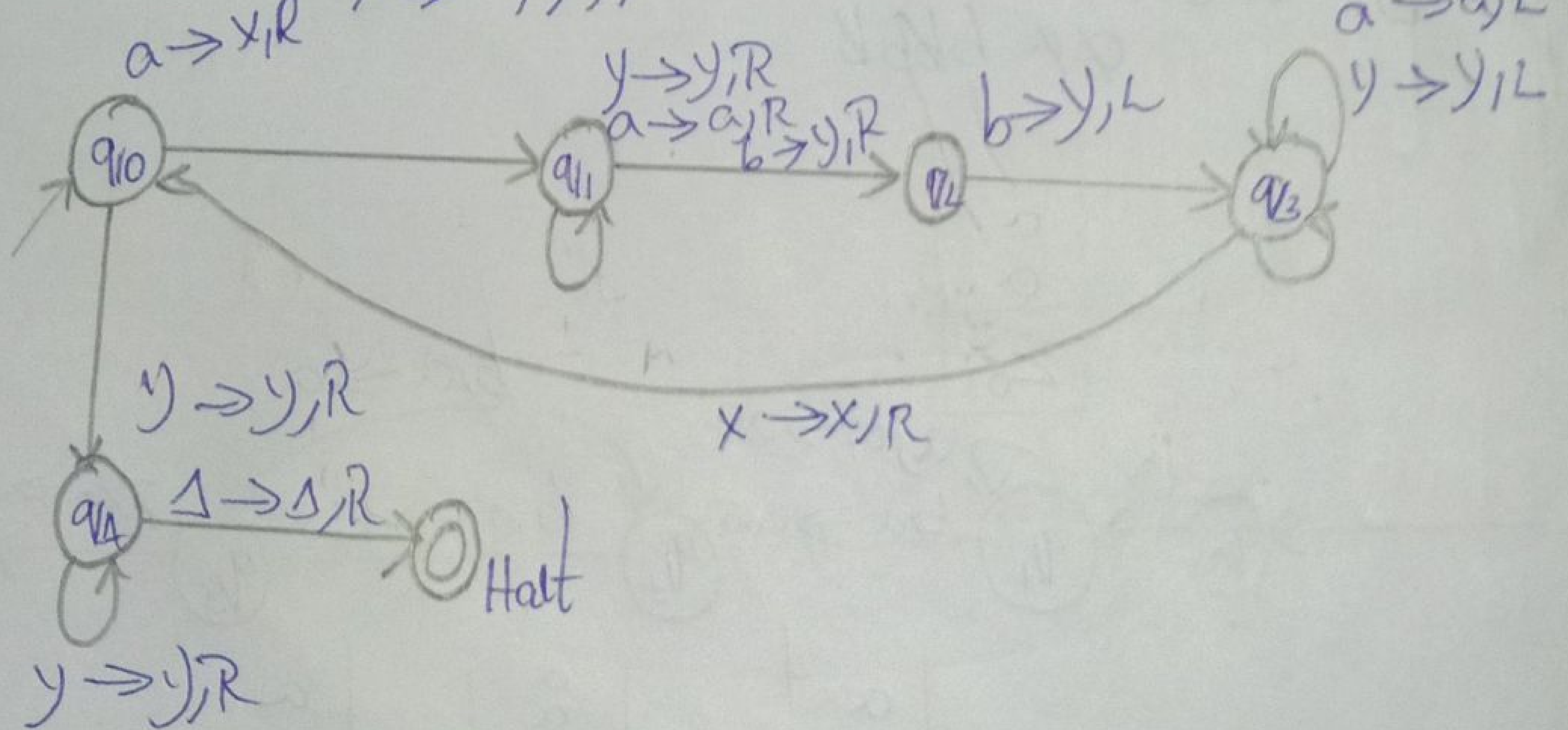
(16) Design TM to accept the input string $A^n B^{2n}$

aa bb bb

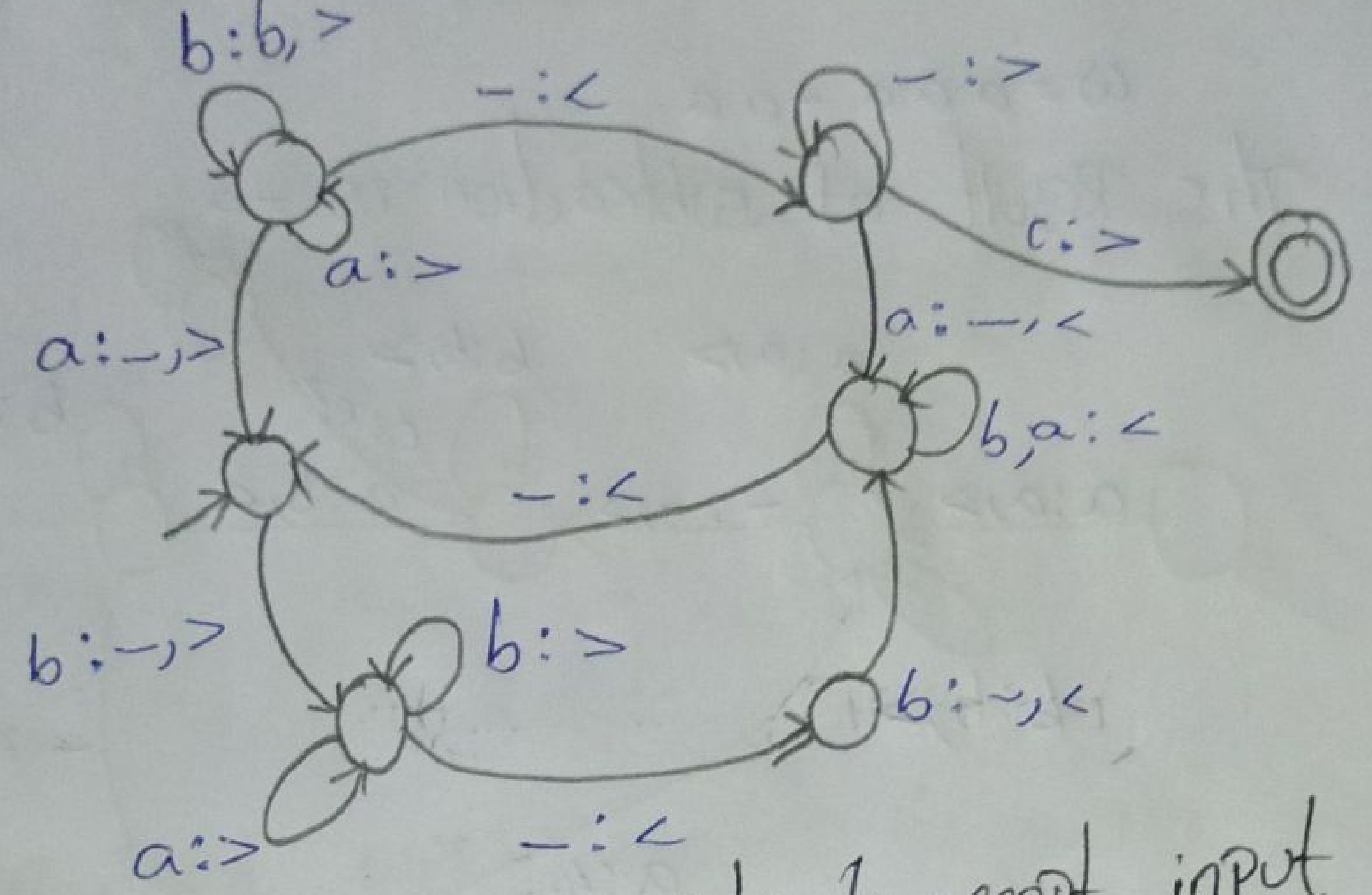
xa yy ←

x yy yy ←

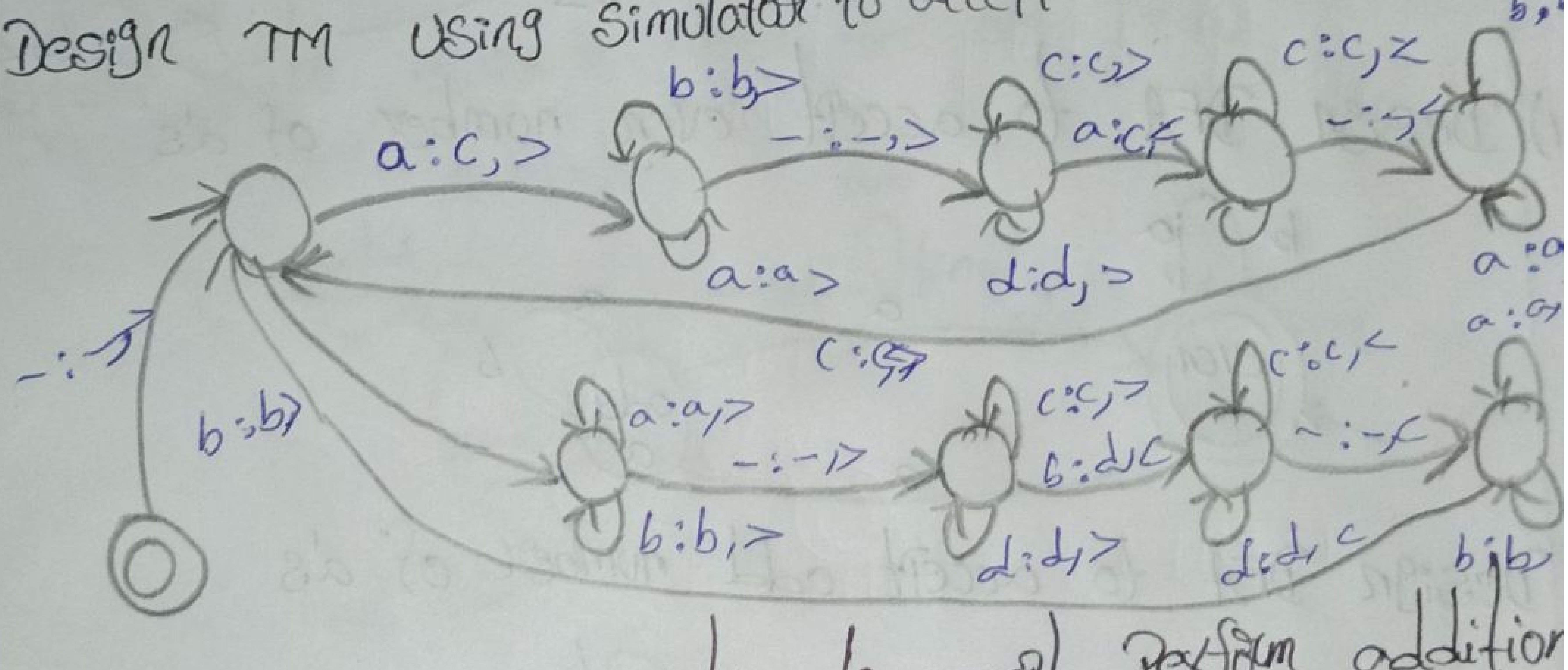
x → yyyy



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



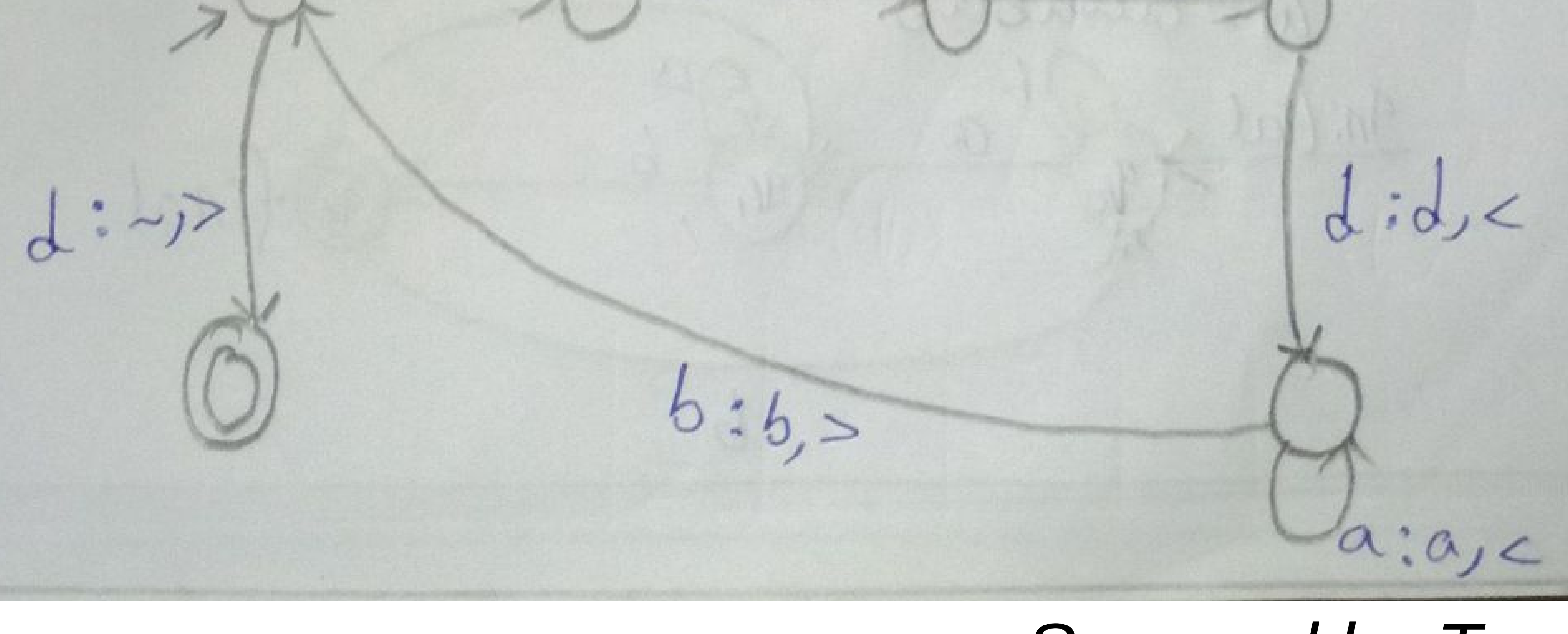
(18) Design TM using Simulator to accept input string ww



(19) Design TM using simulator to accept Perform addition of 'aa' & 'aaa'

W = aa + aaaa

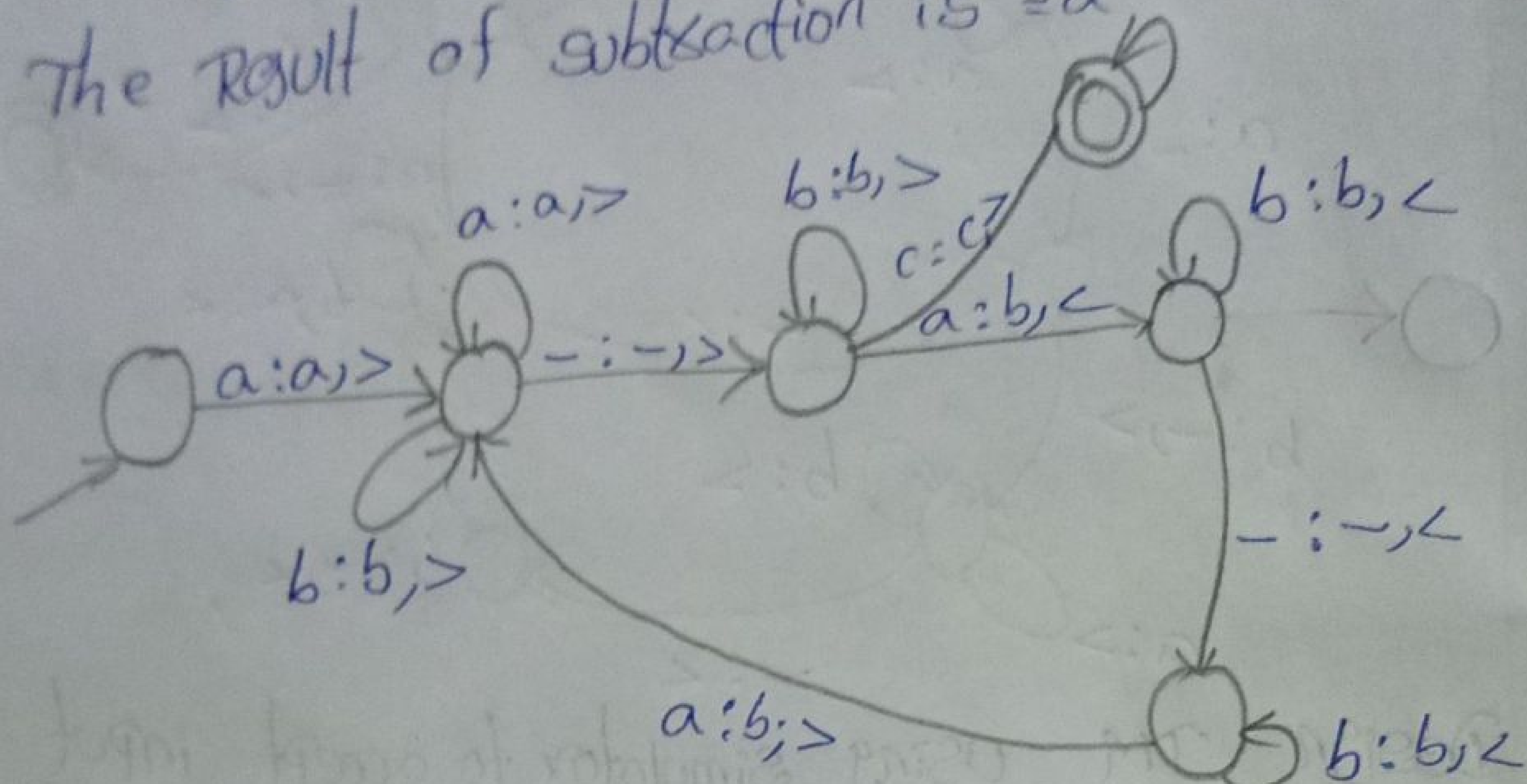
After Addition of a's = aaaaaa



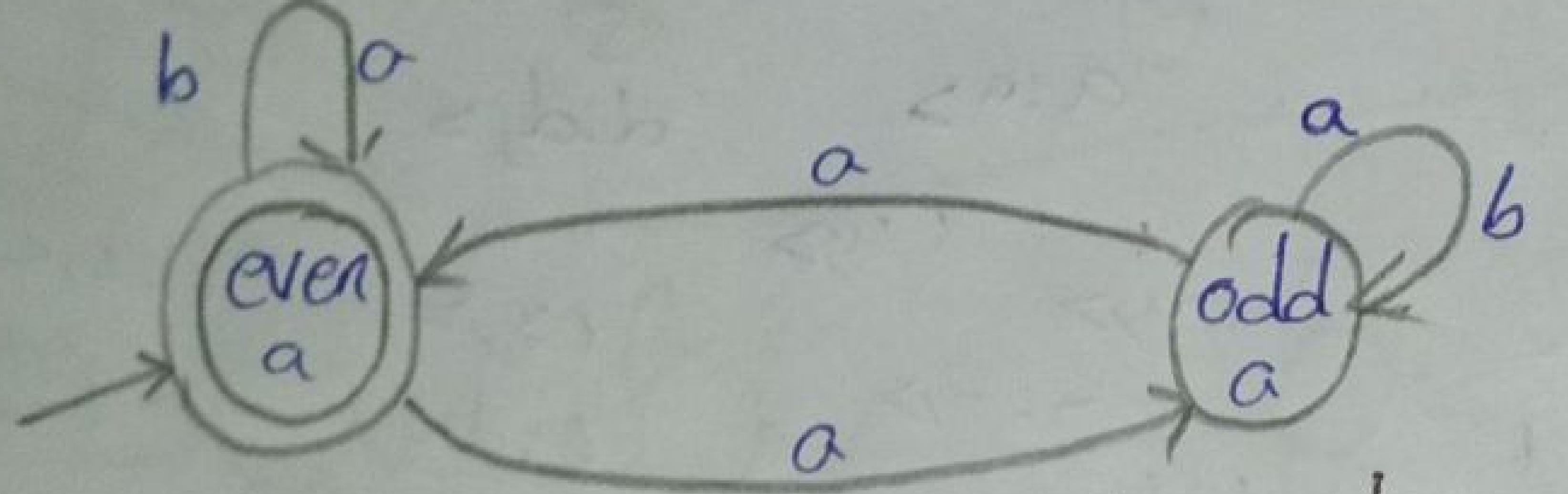
(20) Design TM using simulator to Perform subtraction of 'aaa' and 'aa'

$\omega = aaa - aa$

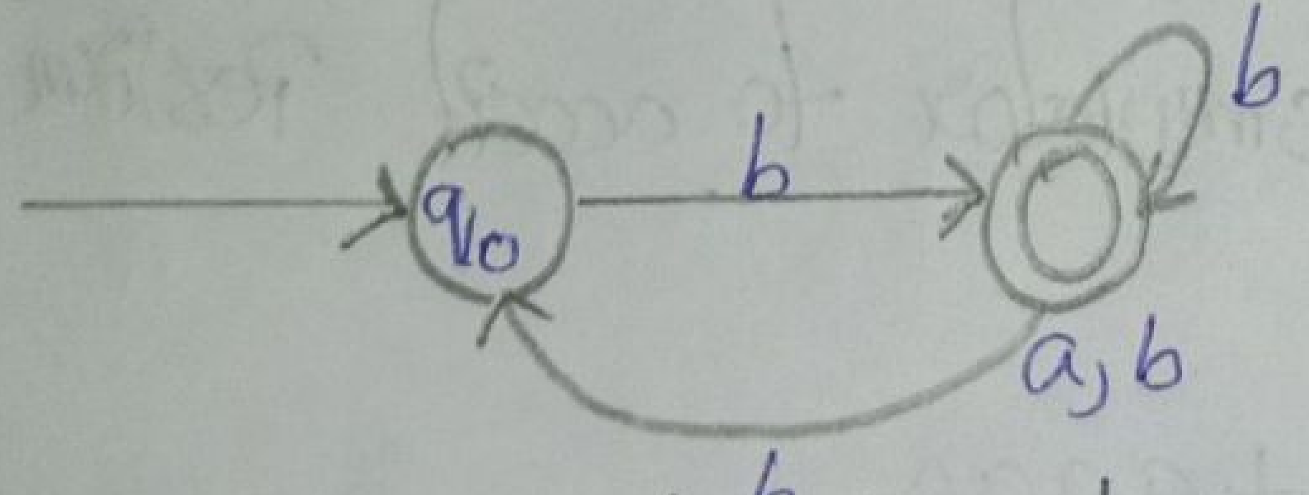
The Result of subtraction is $= a$



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

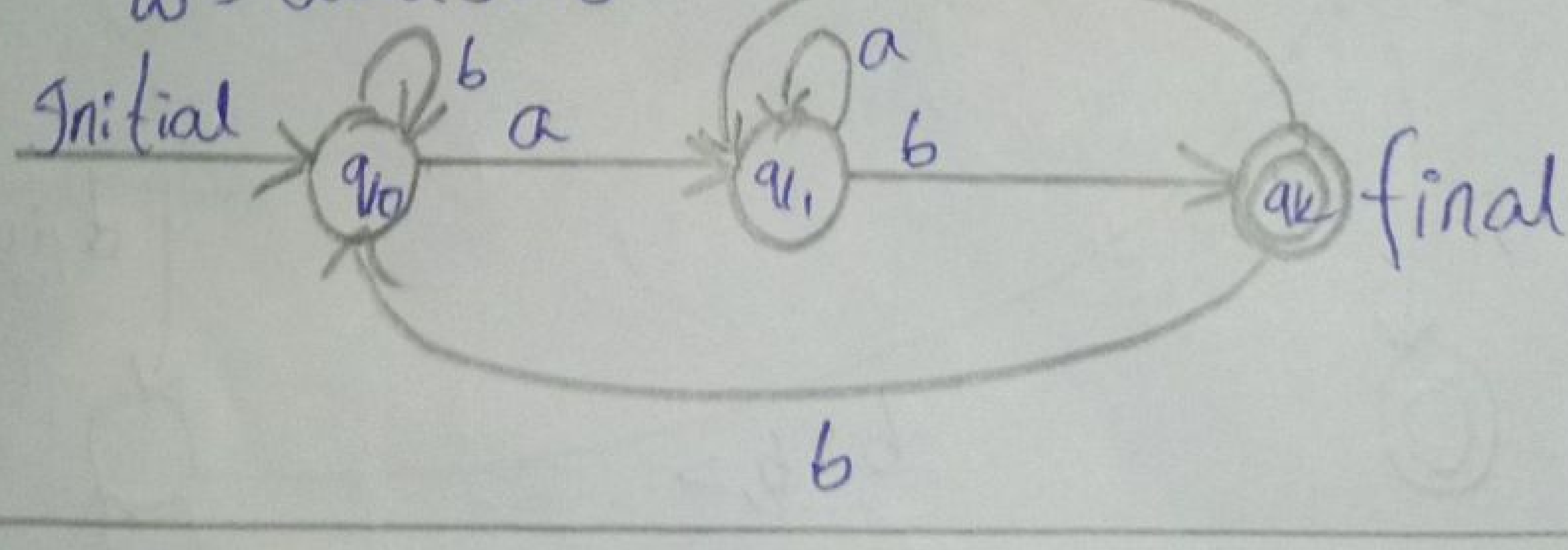


(22) Design DFA to accept odd number of a's

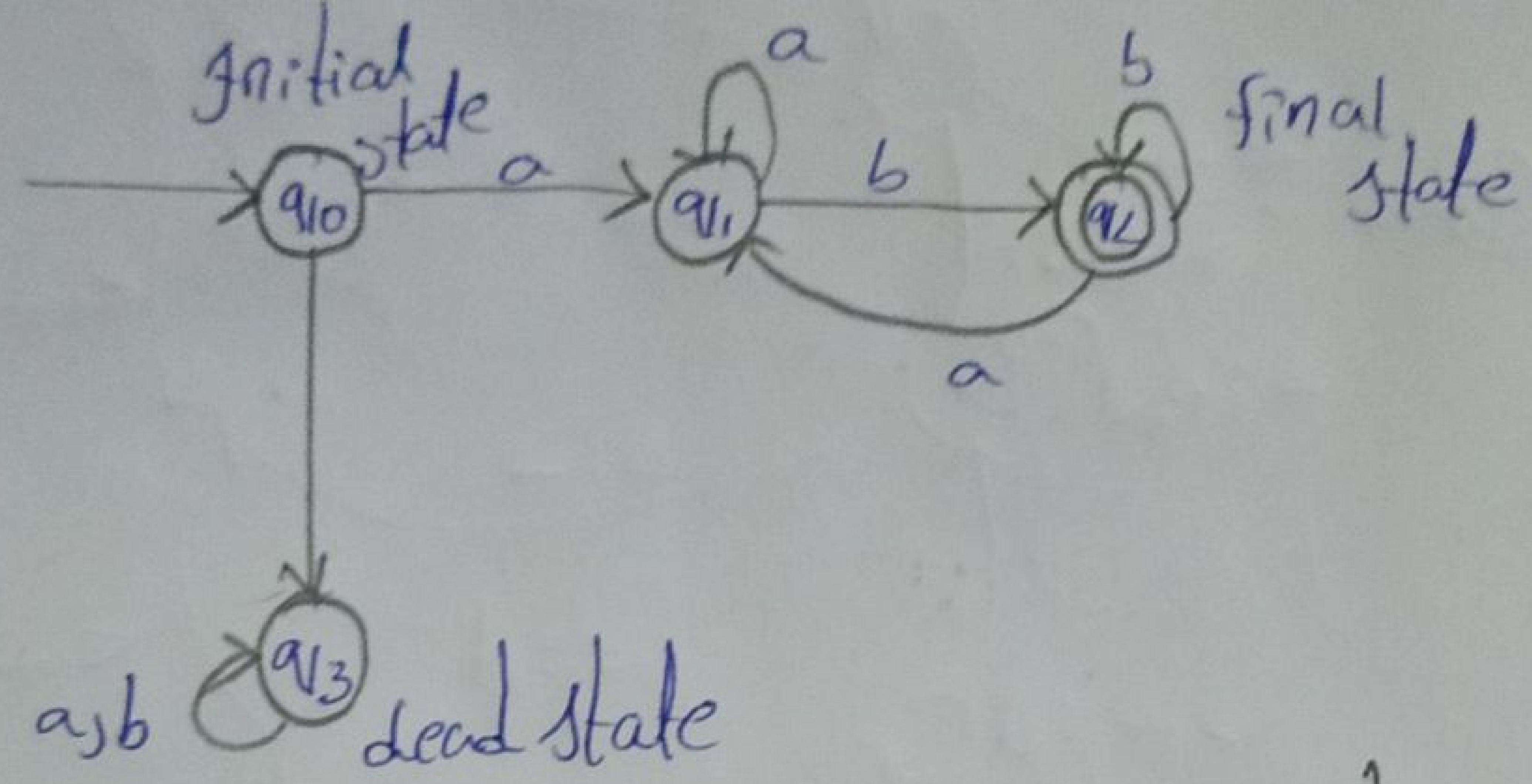


(23) Design DFA to accept the string that with ab over $\{a,b\}$

$\omega = aaabab$



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



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