**CS3062 Theory of Computing**

**Assignment 2**

**P.Thineshan**

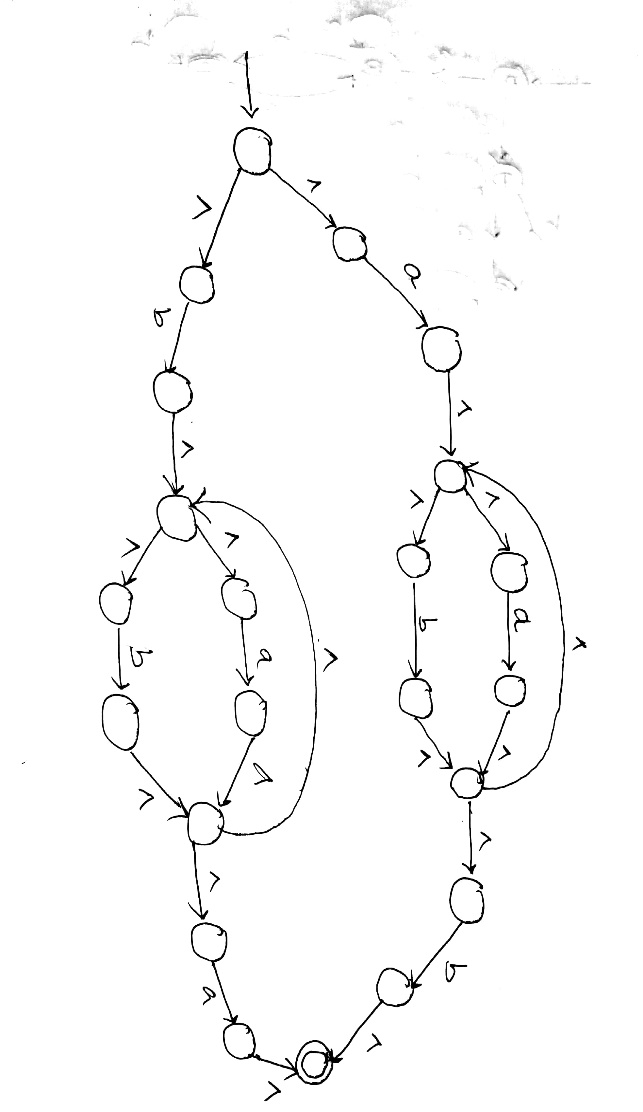
**160633A**

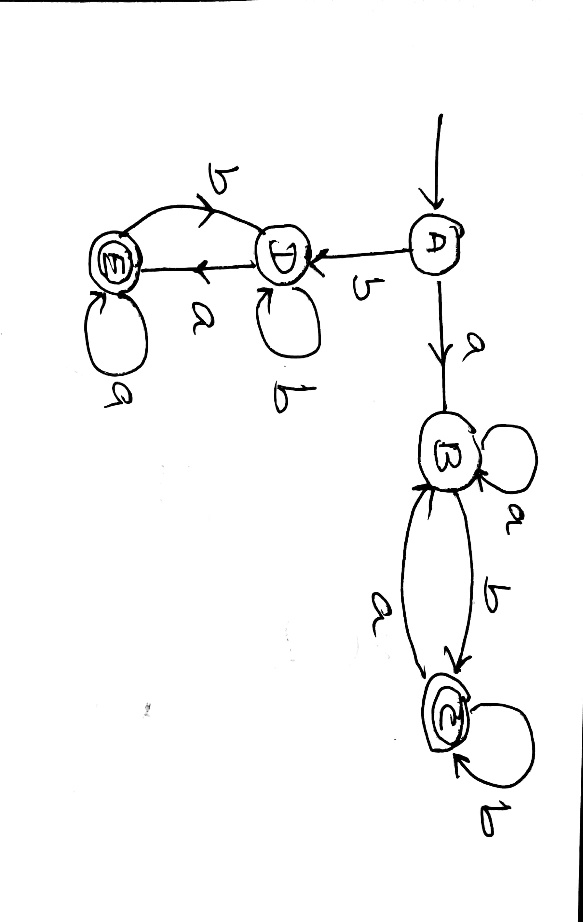
**1.** L is the language over {a, b} such that, for every string in L, if it starts with a then it ends with b and if it starts with b then it ends with a.

(a) Give a regular expression that represents the language L.

a(a+b)\*b + b(a+b)\*a

(b) Construct NFA-Λ for the above expression using thompson’s construction.



(c) Provide DFA corresponding to this language. 

(d) Minimize the states of above DFA.

**2.** S → Ab | aaB, A → a | Aa, B → b

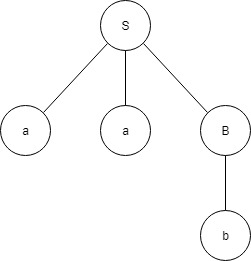
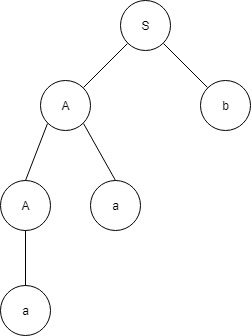
(a) S → Ab S → aaB

S → Aab S → aab

S → aab

string ‘s’ that has at least two leftmost derivations = aab

(b) Two derivation trees for the string ‘s’

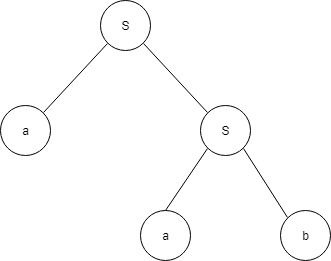
 

(c) Equivalent unambiguous context free grammar

S → aS | ab

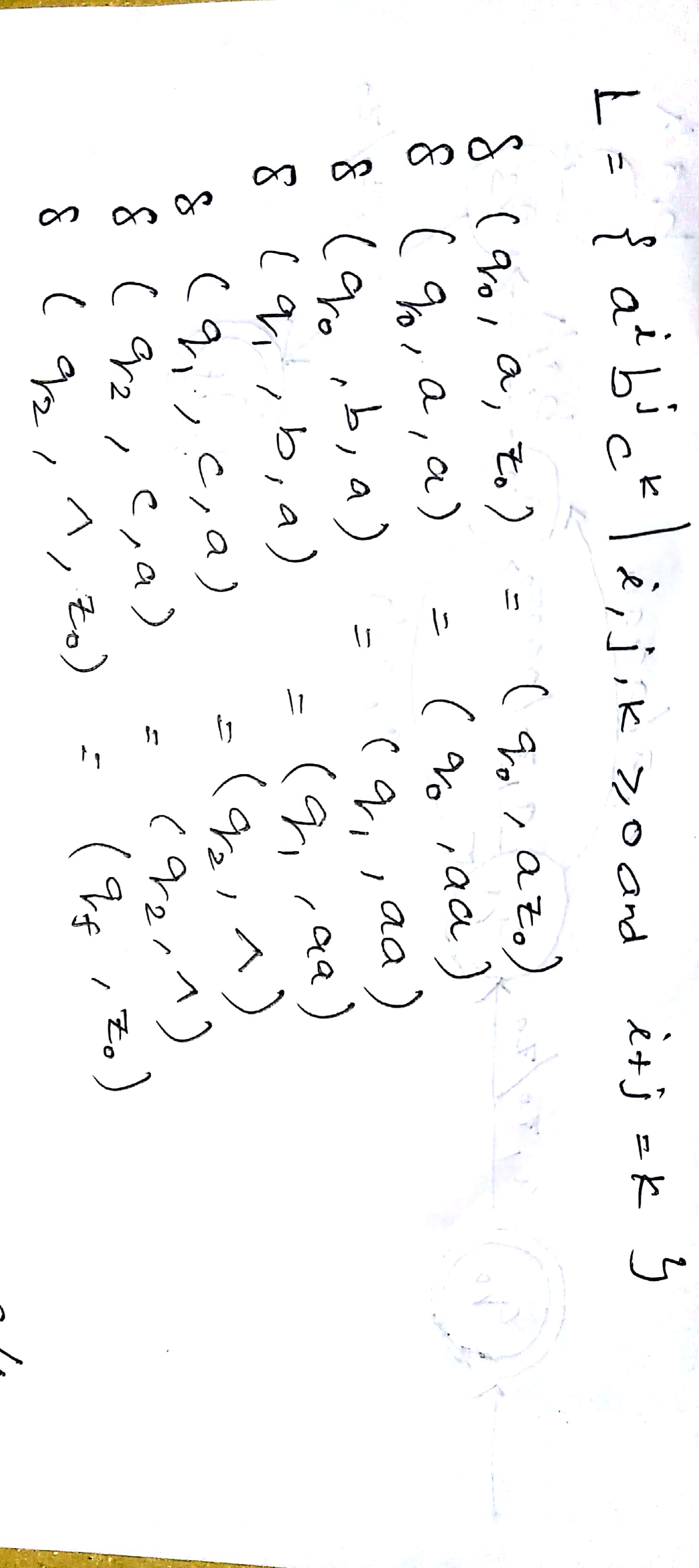
(d) S → aS

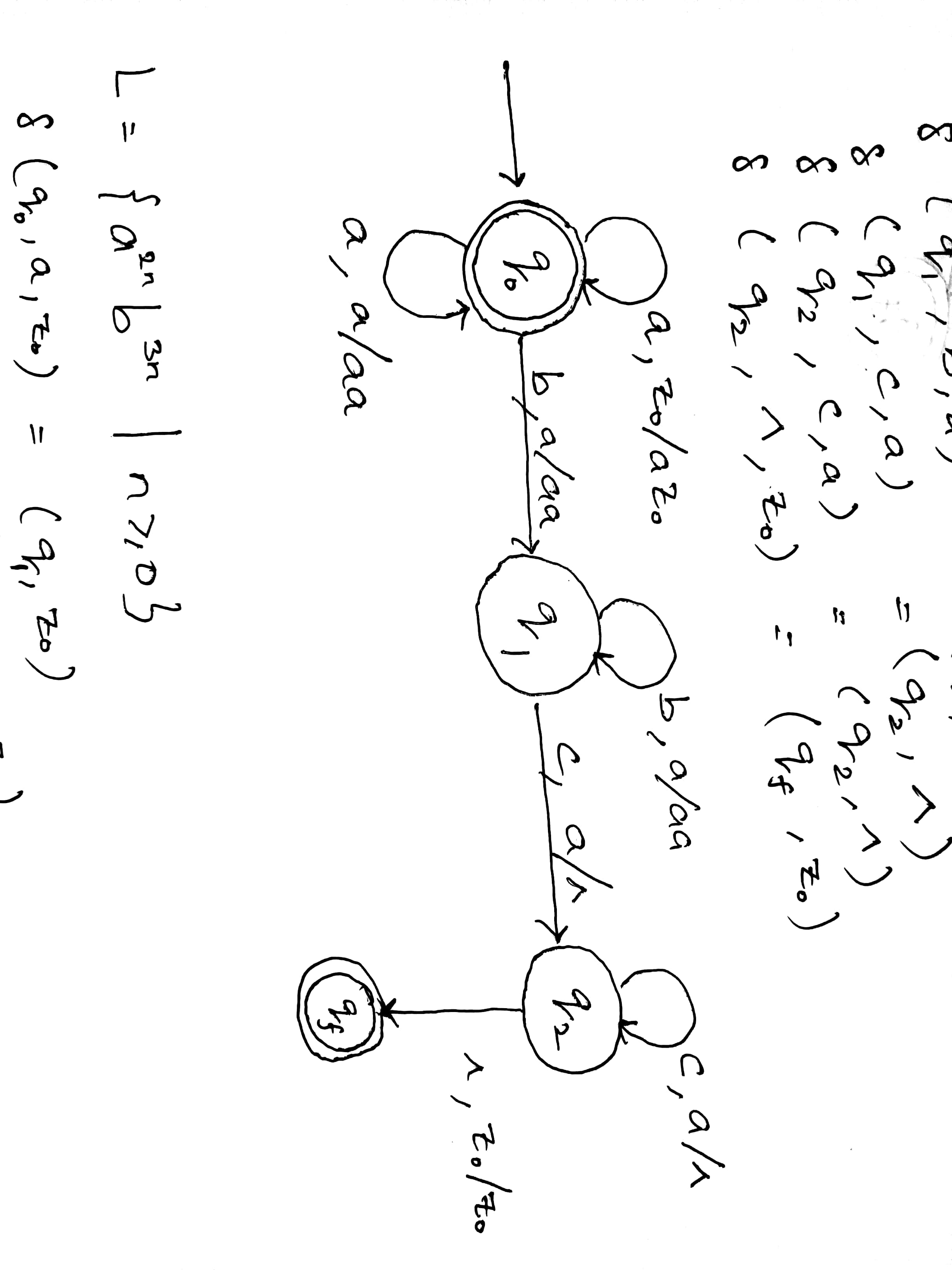
S → aab



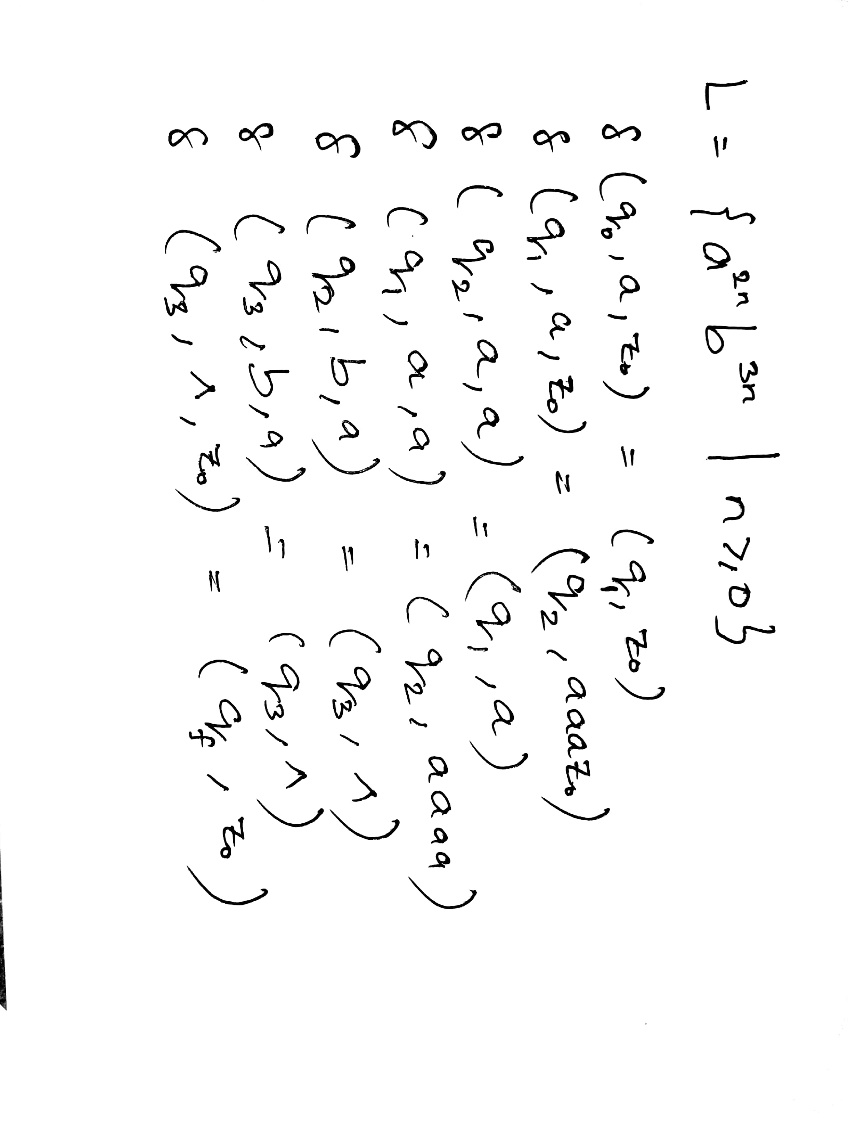
**3.** Construct a PDA (non-determinism is allowed) for each of the following languages.

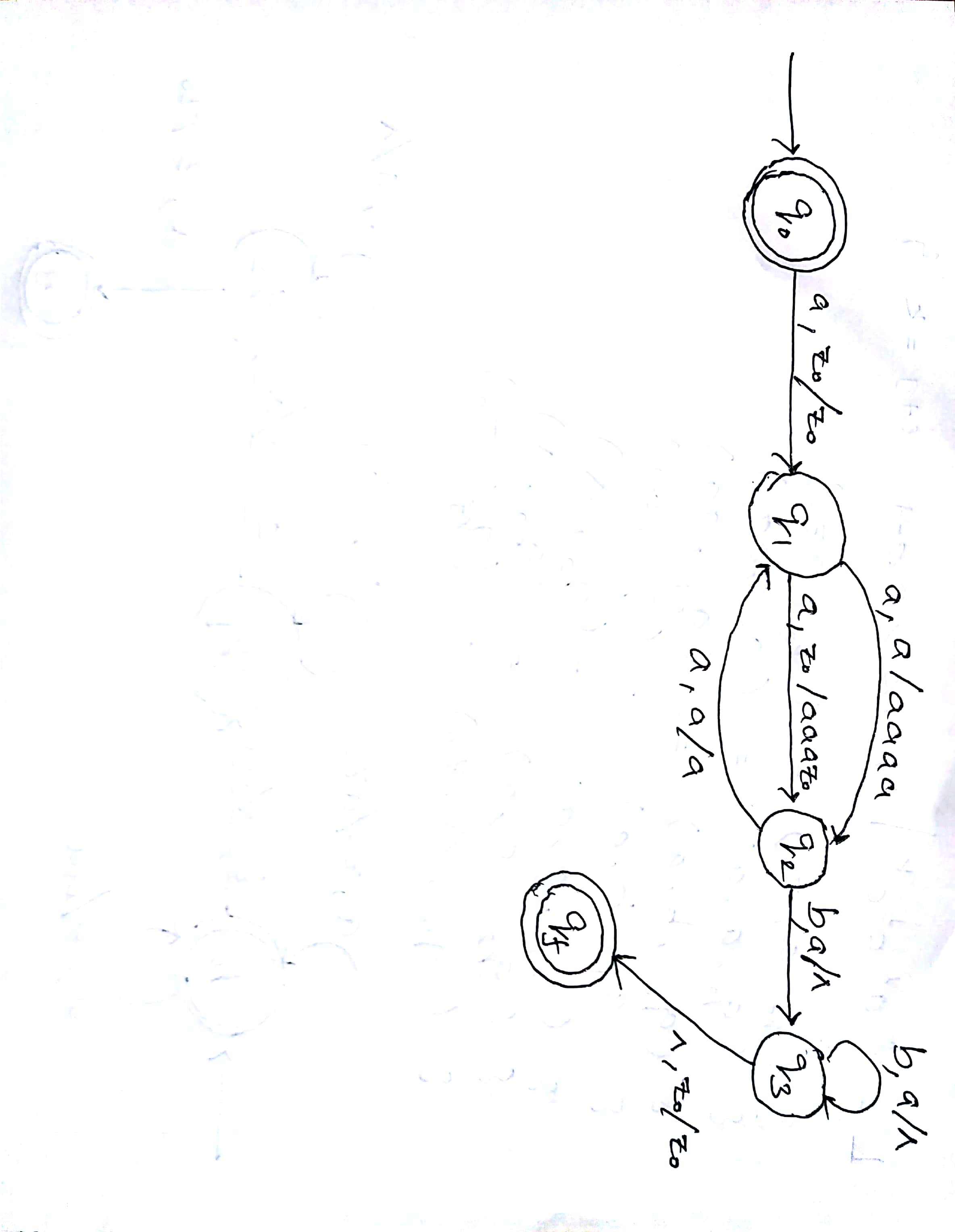
(a)





(b)





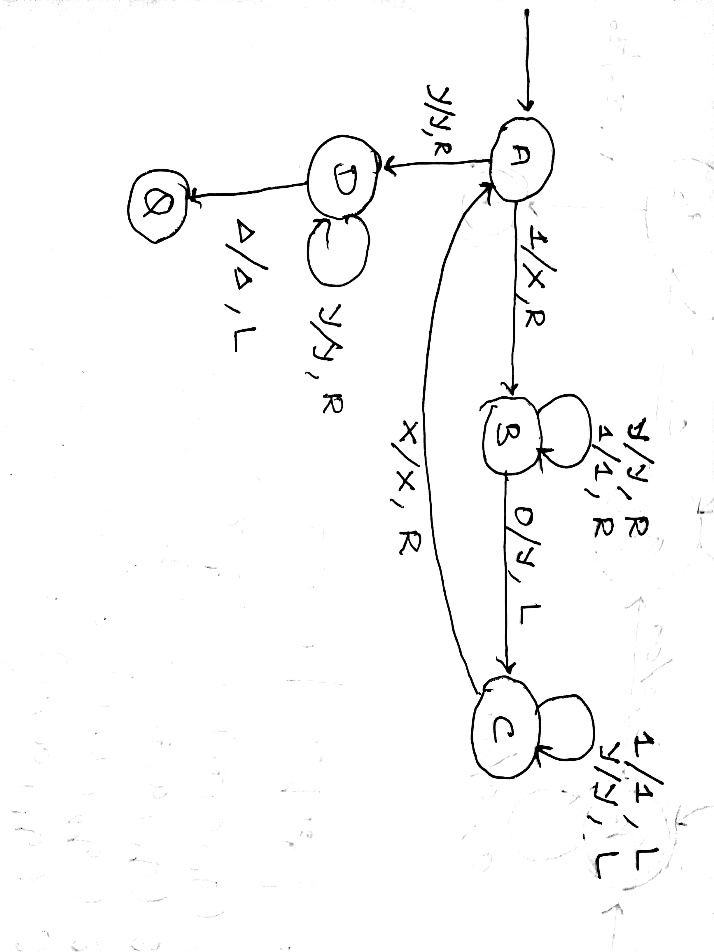
**4.** Design a Turing Machine that decides the language L := {1n 0n | n>= 1} .

**Assumption:** We will replace 1 by X and 0 by Y

**Approach used:**   
First replace a 1 from front by X, then keep moving right till you find a 0 and replace this 0 by Y and move left. Now keep moving left till you find a X. When you find it, move a right, then follow the same procedure as above.

A condition comes when you find a X immediately followed by a Y. At this point we keep moving right and keep on checking that all 0’s have been converted to Y. If not then string is not accepted. If we reach ∆ then string is accepted.

* **Step-1:**  
  Replace 1 by X and move right, Go to state B.
* **Step-2:**  
  Replace 1 by 1 and move right, remain on same state  
  Replace Y by Y and move right, remain on same state  
  Replace 0 by Y and move right, go to state C.
* **Step-3:**  
  Replace 1 by 1 and move left, remain on same state  
  Replace Y by Y and move left, remain on same state  
  Replace X by X and move right, go to state A.
* **Step-4:**  
  If symbol is Y replace it by Y and move right and Go to state D  
  Else go to step 1
* **Step-5:**  
  Replace Y by Y and move right, remain on same state  
  If symbol is ∆ replace it by ∆ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q



**5.**

For each part identify whether the given language L is context-free or non-context-free language. Prove your answer.

a. L = {an bj | n≤j2 and n, j ∊ Z}

b. L = {an | n is prime}

c. L = {an bj ck | k= j\*n and n, j, k ∊ Z}

(a) L = {an bj | n≤j2 and n, j ∊ Z}

**This is not context-free**

Assume for contradiction that L is a context-free language. We apply the pumping lemma. Let m be the parameter of the pumping lemma. We choose to pump the string bm 𝜖L. We have that bm = uvxyz, with |vxy|<= m and |vy|>=1.

We examine all the possible cases for the position of string vxy. First we note that the string v cannot span simultaneously both and bm , since if we pump up v (repeat v), the resulting string is not in the language (a's are mixed with b's). Therefore, it must be that v is either within or within bm . The same holds for y. Below are the rest of the cases. Notice that in all cases we obtain a contradiction, and therefore the language **L is not context-free**.

1. v is within and y is within bm . We have that v = ak and y = bl , with 1 <=k + l <= m (since |vxy| <= m and |vy|>=1). Consider the case where l >=1. From the pumping lemma we have that uv0xy0z 𝜖L. Therefore, bm-l 𝜖L, and thus, it must be that m2-k <= (m-l)2 . However, this is impossible since:

(m-l)2 <= (m-l)2 (since l>=1)

= m2 +2m + 1

< m2-k (since k <= m)

Consider now the case where l = 0. It must be that k >= 1 (since k + l >= 1). From the pumping lemma we have that uv2xy2z 𝜖 L. Therefore, bm 𝜖 L, which is impossible since m2+k>m2

1. v and y are within . If we pump up v and y(repeat them) we obtain a string of the form , bm, with k>=1, which obviously is not in the language
2. v and y are within bm . If we pump down v and y (remove them) we obtain a string of the form bm-k , with k >= 1, which obviously is not in the language.

(b) L = {an | n is prime}

**This is not context-free**

Assume L is a context-free language. We apply the pumping lemma. Let m be the parameter of the pumping lemma. Let p be a prime such that p >= m.

We choose to pump the string ap𝜖L. Since ap = uvxyz, we have that v = ak and y = al , with k+l >= 1 (since |vy|>=1). From the pumping lemma we have that uv1+pxy1+pz 𝜖 L, and therefore ap+kp+lp 𝜖 L. Subsequently, ap(1+k+l) 𝜖 L., which is impossible since p(1+k+l) is not a prime. Thus, we have a contradiction and the language **L is not context-free**.

(c) L = {an bj ck | k= j\*n and n, j, k ∊ Z}

**This is not context-free**

Assume L is a context-free language. We apply the pumping lemma. Let m be the parameter of the pumping lemma. Wechoose to pump the string ambm 𝜖 L. We have that ambm = uvxyz, with |vxy|<= m and |vy|>=1.

We examine all the possible cases for the position of string vxy. First we note that the string v cannot span simultaneously both am and bm, since if we pump up v (repeat v), the resulting string is not in the language (a's are mixed with b's). Similarly, v cannot span both bm and . Therefore, it must be that v is either within am or bm or . The same holds for y. Below are the rest of the cases. Notice that in all cases we obtain a contradiction, and therefore the **language L is not context-free**.

1. v is within bm and y is within. We have that v = bk and y = cl , with 1 <=k + l <= m (since |vxy| <= m and |vy|>=1).

Consider the case where k >=1. It must be that l<m (since k+l<=m). From the pumping lemma we have that uv0xy0z 𝜖L. Therefore, ambm-k 𝜖L, and thus, it must be that m.(m-k) = m2 - l . However, this is impossible since:

m.(m-k) = m2 -mk

<= m2 -m (since k>=1)

< m2-l (since l < m)

Consider now the case where k = 0. It must be that l >= 1 (since k + l >= 1). From the pumping lemma we have that uv0xy0z 𝜖 L. Therefore, ambm 𝜖 L, which is impossible since m.m not equal m2 -l

1. v and y are within bm .

If we pump down v and y (remove them) we obtain a string of the form ambm-k, with k >= 1, which obviously is not in the language.

1. v and y are within . If we pump up v and y(repeat them) we obtain a string of the form , ambm-k, with k>=1, which obviously is not in the language
2. v and y are somewhere within ambm

Similar to cases (ii) and (iii)