Chapter 1
Mathematics Preliminary

Answer for Exercises (Page 13, Theory of Computer Science: Definitions and Examples. 3rd Ed.)

1.

a) {1,10,100}
b) { n | $n \in$ Z and $n > 5$ }
c) { n | $n \in$ N and $n < 5$ }
d) {*aba*}
e) { ε } or { λ }
f) Ø

2.

a) A union B, which consists of x, which is member of A or x is member of B.
b) A intersect B, which consists of x, which is member of A and x is member of B.
c) A minus/differ B, which consists of x, which is member of A and x is not member of B.
d) A minus/differ B, which consists of x, which is member of A or x is member of B and x is not member of both A and B.
e) x is not member of A.
f) cardinatility of set A. Length or no. of elements in A.
g) powerset of set A

3.

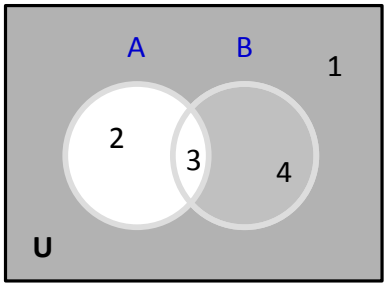
a) Is A a subset of B? FALSE.
b) Is B a subset of A? TRUE.
c) What is A ∪ B? Answer: A
d) What is A ∩ B? Answer: B
e) What is A x B? Answer: {(x,x), (x,y), (y,x), (y, y), (z, x), (z, y)}
f) What is the power set of B? Answer: { Ø, {x},{y},{x,y}}

4.

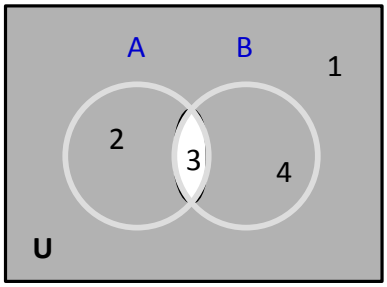
a) {0, 6, 7, 8, 9}
b) {0, 1, 2, 3, 5, 6, 7, 8, 9} = X' = {0, 1, 5, 6, 7, 8, 9}, Y' = {0, 2, 3, 6, 7, 8, 9}, X' U Y' =.
c) {2, 3, 4, 5}
d) {2, 5}

5.

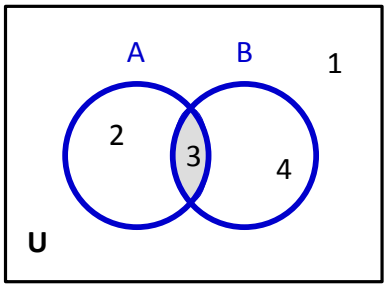
a)



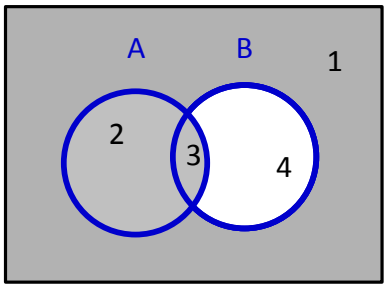
b)



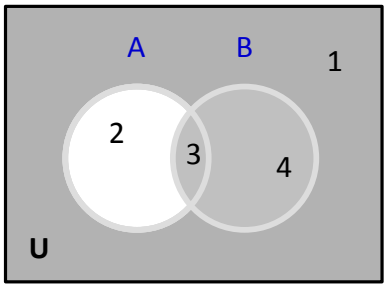
c)



d)



e)



6.

a) {0, 1, 2, 3, 4, 6}
b) {2}
c) {1, 3}
d) {0, 4, 6}

7.

a) {{λ}, {a}, {b}, {a,b}}

b) {a1, a2, a3, b1, b2, b3}

8.

a) {{a, b}, {{a, b}}}
b) {a, b, d}
c) {{}{1}{2}{3}{1,2}{1,3}{2,3}{1,2,3}}-{{} {1}{3}{1,3}} = {{2}{1,2}{2,3}{1,2,3}}
d) {{}{a}{b}{a,b}} X {{}{c}{d}{c,d}} = {({}, {}),({}{c}),({}{d}),({}{c,d}) … ({a,b}, {}),({a,b}{c}),({a,b}{d}),({a,b}{c,d})}
e) {{}}
f) {({}, a), ({}, b), ({a}, a), ({a}, b), ({b}, a), ({b}, b), ({a, b}, a), ({a, b}, b)}
g) {(1,1,1), (1,1,2), (1,1,3), (1,2,1), (1,2,2), (1,2,3)}

9. 5

10. A x B has a*b elements. A x B stands for cartesian product which is formed as set of tuples taking each element from each set.
So for 2 x 2 set. {a, b} x {c, d} = { (a,c), (a,d), (b,c), (b,d)}, thus there are 4 elements.

11. There are x^2 elements in the power set of X. In order to explain this conclusion, I ask you to imagine a string of x bits, each bit representing an element in X. If we allow 0 to indicate the absence of an element in the superset and 1 to indicate the presence of an element, we'll find the number of combinations of 0s and 1s to be the number of elements in the powerset. For a binary number of length x, we have x^2 such combinations.

12.

a) It is the set of all odd natural numbers.
b) It is the set of all even real numbers.
c) It is set of even natural numbers.
d) It is set of natural numbers which are divisible by both 2 and 3.
e) It is set of binary numbers which are bi-directional (that is read the same from left to right and also from right to left).
f) It is set of all integers.

13.

A’ = Everything in set U but NOT in set A = {18, 19, 20, 22, 24, 26, 28}
B' = Everything in set U but NOT in set B = {18, 19, 20, 21, 22, 23, 24, 25}

Now simply combine the two sets A' and B' (and remove any duplicate elements) to get

A’ U B’ = {18, 19, 20, 22, 24, 26, 28} U {18, 19, 20, 21, 22, 23, 24, 25}
= {18, 19, 20, 21, 22, 23, 24, 25, 26, 28}

So the answer is A’ U B’ = {18, 19, 20, 21, 22, 23, 24, 25, 26, 28}

14.

U = {l, m, n, o, p, q, r, s, t, u, v, w}
A = {l, m, n, o, p, q}
B = {n, o, r, s, v, w}
C = {l, m, p, q, r, t}
find (A′ U C′) ∩ B′.
A′ = {r,s,t,u,v,w}
C′ = {n,o,s,u,v,w}
A′ U C′ = {n,o,r,s,t,u,v,w}
B′ = {l,m,p,q,t,u}
(A′ U C′) ∩ B′ = {t,u}

15.

a) A ∩ B = Set of elements that BOTH sets A and B have in common
A ∩ B = {1, 2, 3} ∩ {3, 4, 5, 6}
A ∩ B = {3}

b) A ∩ C = Set of elements that BOTH sets A and C have in common
A ∩ C = {1, 2, 3} ∩ {3, 5, 7}
A ∩ C = {3}

c) A U C = Combination of the sets A and C (remove any duplicates)
A U C = {1, 2, 3} U {3, 5, 7}
A U C = {1,2,3,5,7}

d) B U C = Combination of the sets B and C (remove any duplicates)
B U C = {3, 4, 5, 6} U {3, 5, 7}
B U C = {3,4,5,6,7}

16.

To start with

- I am going to assume that U = {All the lowercase letters of the alphabet}. If this is incorrect, then the answers to (c), (d) and (e) will be incomplete.
- To avoid any potential confusion, I will use U for the "universal" set and "u" for the union operator
- When the elements of a set are listed, the order is not important
- The union operator, u, is an inclusive operator. The union of two sets will always include every member of both sets. Elements of both sets, if any, are listed only once in the union. (Never list duplicate elements in any set.)
- The intersection operator, ∩, is an exclusive operator. The intersection of two sets will include only elements that are members of both sets.

(a) A u B
This means "the set whose elements are members of set A or set B (or both)".
A u B = {c, d, e, f, g, h, k}

(b) A ∩ B
This means "the set whose elements are members of both set A and set B".
A ∩ B = {e, f}

(c) A′ ∩ B′
A′ means "the set of all elements of the universal set, U, which are not members of set A.
A′ = {a, b, g, h, i, j, k, l, m, ..., z}
B′ means "the set of all elements of the universal set, U, which are not members of set B.
B′ = {a, b, c, d, i, j, l, m, ..., z}

A′ ∩ B′ means "the set whose elements are members of both set A' and set B'".
A′ ∩ B′ = {a, b, i, j, l, m, ..., z}

(d) A′ u B′
This means "the set whose elements are members of set A′ or set B′ (or both)". Using A′ and B′ from (c) we get
A′ u B′ = {a, b, c, d, g, h, i, j, k, l, m, ..., z}

(e) A u B′
This means "the set whose elements are members of set A or set B′ (or both)". Using A and B′ from (c) we get
A u B′ = {a, b, c, d, e, f, i, j, l, m, ..., z}

(f) (A u B′) ∩ B
The first part of this is the answer to part (e). (A u B′) ∩ B means "the set whose elements are members of the set B and members of the answer to part (e)"
(A u B′) ∩ B = {e, f}

(g) (A u B) ∩ (A u B′)
The first part is the answer to part (a), The last part is the answer to part (e). So (A u B) ∩ (A u B′) means "the set whose members are members of the answer to part (a) and members to the answer to part (e)". So
(A u B) ∩ (A u B′) = {c, d, e, f} (which is set A}

17.

a) $A \bigcup B$ is simply the union between the two sets. So simply take all of the uniqe elements of each set and put them together in the new set $A \bigcup B$

So $$A \bigcup B = \{b,c,d,e,f,g\}$$

b) $A \bigcap B$ is simply the intersection between the two sets. So whatever elements in two sets have in common will make up $A \bigcap B$ (which in English says "set A intersect with set B")

Since set A and set B have the element "c" in common, this means the intersection of sets A and B gives us:

$$A \bigcap B = \{c\}$$

c) The notation A' simply means the set of every letter that is NOT in set A. So A' is the entire alphabet but with the letters b,c, and d taken out of it

So $$A' = \{a,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

The notation B' simply means the set of every letter that is NOT in set B. So B' is the entire alphabet but with the letters c, e, f, and g taken out of it

So $$B' = \{a,b,d,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

Now let's take the intersection between the two sets.

Since set A' and set B' have the elements a, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, and z in common, this means the intersection of sets A' and B' gives us:

$$A' \bigcap B' = \{a,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

d) Using the same sets A' and B' from part c)

$A' \bigcup B'$ is simply the union between the two sets. So simply take all of the uniqe

elements of each set and put them together in the new set $A' \bigcup B'$

So

$$A' \bigcup B' = \{a,b,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

Start with the given sets:

$$A = \{b,c,d\}$$

$$B = \{c,e,f,g\}$$

Find their respective complement sets:

$$A' = \{a,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$
$$B' = \{a,b,d,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

e) So

$$A \bigcup B' = \{b,c,d,a,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

Remember, a union is simply the combination of the unique elements from both sets

f) From part a), we got

$$A \bigcup B' = \{b,c,d,a,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$$

So

$$\left(A \bigcup B'\right) \bigcap B = \{b,c,d,a,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\} \bigcap \{c,e,f,g\} = \{c\}$$

Remember, to find the intersection of two sets, simply look for any elements that the two sets have in common

g) First find $A \bigcup B$

$$A \bigcup B = \{b,c,d,e,f,g\}$$

So

$$\left(A \bigcup B\right) \bigcap \left(A \bigcup B'\right) = \{b,c,d,e,f,g\} \bigcap \{b,c,d,a,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\} = \{b,c,d\}$$

18.

To find the number of elements in the set K ∪ J, simply add the two sets together. However, remember to subtract the number of elements in both sets (since these elements are repeated)

In other words, use this formula:
$n(K \cup J) = n(K) + n(J) - n(K \cap J)$

In this case, $n(K)=30$, $n(J)=46$ and $n(K \cap J) = 11$. Plug these values in to get:
$n(K \cup J) = 30 + 46 - 11$

Add $n(K \cup J) = 76 - 11$
Subtract $n(K \cup J) = 65$
So there are 65 elements in set K ∪ J
$\#(K \cup J) = \#(K) + \#(J) - \#(K \cap J) = 30 + 46 - 11 = 65$

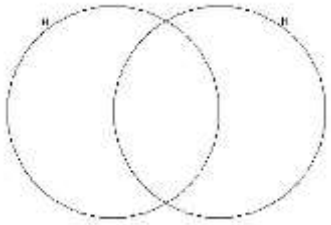
19.

$$A = \{x | x \in \mathbb{N} \quad and \quad 13 < x < 18\}$$
$$A = \{14,15,16,17\}$$
$$B = \{12,13,14,15,16,17,18\}$$

Notice how ALL of the elements of set A are also elements of set B, so this shows us that set A is a subset of set B (ie all of A is in B). So $A \subseteq B$

Note: the two sets A and B are NOT equal since set B has the element 12 (but set A does NOT have the element 12)

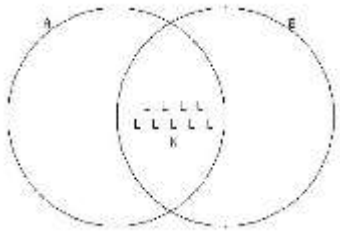
20. Draw two overlapping circles, label one A and one B, like this



First look at these words:

9 letters and 1 number is common to both sets.

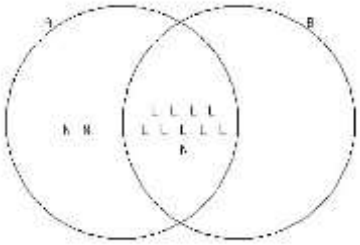
Letting L stand for a letter and N stand for a number, put 9 L's and 1 N in the part in the middle which is common to both circles, like this:



Next look at these words:

Set A has 9 letters and 3 numbers.

Set A already has 9 letters and 1 number because they are in the part that is in common to both circles. So set A needs 2 more numbers so it will have 3 numbers. (It doesn't need any more letters). So we put 2 N's in the left part of A, like this:

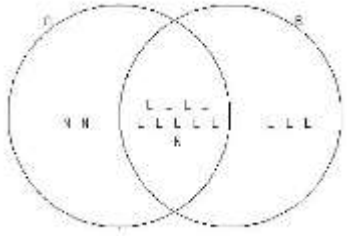


and now set A has 9 letters and 3 numbers.

Next we look at these words:

Set B has 12 letters and 1 number.

Set B already has 9 letters and 1 number because they are in the part that is in common to both circles. So set B needs 3 more letters so it will have 12 letters. (It doesn't need any more numbers). So we put 3 L's in the right part of B, like this:

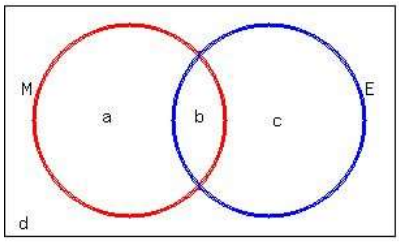


and now set B has 12 letters and 1 number.

Now the question is:
What is the number of elements in set A or set B?
If we count all the L's and N's that are in either one of the two sets, we see there are 12 L's and 3 N's, so that makes 15 elements.

Answer: 15 elements.

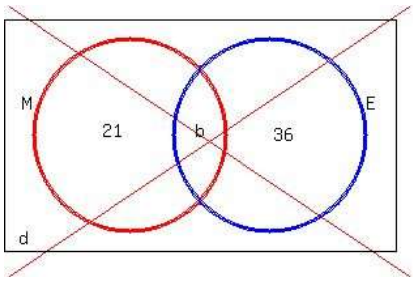
21. In a survey of a college with 50 members, 21 were taking mathematics, 36 were taking English, and 9 were taking both. How many were not taking either of these subjects?
We draw a rectangle with two overlapping circles, one red and one blue.



There are two circles. The red one contains all the students taking math. The blue one contains all the students taking English. Note that the red circle and the blue circle overlap.

Here is the mistake many students make, so DO NOT make this mistake:

Many students would see the words "21 were taking mathematics,36 were taking English,",
and put all 21 in region "a", and all 36 in region "c", like this:

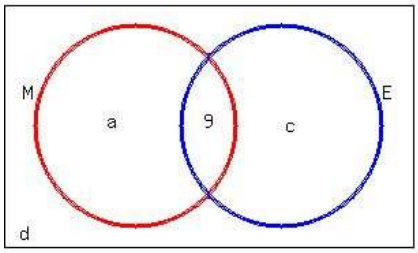


DO NOT make that error!!! Some, but not all, of the 21 go in region "a" and the rest of the go in region "b". Similar, Some, but not all, of the 36 go in region "d" and the rest of the go in region "b".

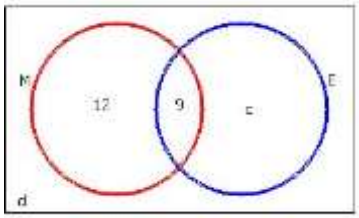
So we do not begin with either the 21 or the 36.

Instead we begin with "9 were taking both". These 9 go in the middle region, b, because those 9 are in both the red circle and the blue circle.

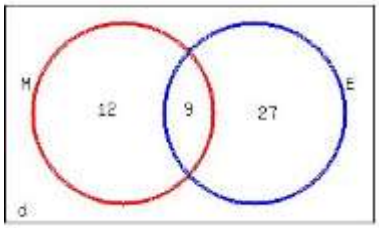
So we put the 9 in region b:



Now we see "21 were taking mathematics", and of this 21, 9 are taking English too, so we must subtract the 9 that are in the middle region, getting 21-9 = 12, and we put 12 in region "a":

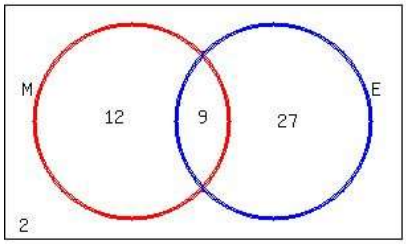


Now we see "36 were taking English", and of this 36, 9 are taking too, so we must subtract the 9 that are in the middle region, getting 36-9 = 27, and we put 27 in region "c":



We have one more region, "d". That is the region outside of both circles. They are the ones that are neither taking mathematics nor taking English.

We have accounted for 12+9+27 = 48. We are told there are 50 members. So there are 50-48 = 2 that have not been accounted for. Those 2 go in the outer region "d":



So the answer to the problem is those 2.

22. F' = Set of everything in U but NOT in F = {Lower cost, Less time to see the sights, Can visit relatives along the way}

D' = Set of everything in U but NOT in D = {Higher cost, More time to see the sights, Cannot visit relatives along the way, Less time to see the sights}

Now just look for the common elements in both sets. Since there are no common elements, this means that the result of the intersection between these sets is the empty set.

23. The first thing is to find out how many do take either language.
Since 25 don't take either that means that 80-25 take at least one of them.
55 take at least one of them
30+40 study French and Spanish.
70-55 take both French and Spanish
and 15 take only Spanish and 25 take only French.

24. First draw 2 overlapping circles call the left circle aquarium and the right zoo.
Second, draw small horizontal lines in each of the three regions and also one on the outside right of the diagram.

step 1 - "15 had both" means put 15 into the overlapping region shared by both.
step 2 - "22 had a membership to the Birch Aquarium" means that 15 + x = 22 - > so x = 7. Put a 7 on the aquarium line.
step 3 - "26 had a membership to the San Diego Zoo" means 15 + y = 26 - -> so y = 11. Put an 11 on the zoo line.
step 4- there is only one line left, the outside. "140 San Diego residents" means that all lines of information must add to 140. We have 7 + 15 + 11 + z = 140 - -> z = 107. Put a 107 on the outside line.

i. How many had only a zoo membership? this is y, so 11

ii. How many had only an aquarium membership? this is x, so 7
iii. How many belonged to either one or the other or both? this is 15 + 7 + 11 = 33
iv. How many belonged to neither? this is z, so 107

26-15=11
11 had only zoo
22-15=7
7 had only aquarium
7+15+11=33
33 belonged to one or both
140-33=107
107 belonged to neither

25. Since 35 students are in algebra, and 18 are in both, this means that 35-18=17 students are in algebra only.

Also, because 52 students are in history, and 18 in both classes, there are 52-18=34 students in just history alone.

Now add up the number of students in algebra only (17), the number of students in history only (34), and the number of students in both (18) to get: 17+34+18=69

So there are 69 students in either class or both. Now subtract this figure from 100 to find the number of students who are in neither class: 100-69=31

So there are 31 students in neither class.

26. Use Venn Diagrams: Draw two intersecting circles inside a rectangle.

Label one circle "brown eyes" ; Label the other circle "blonde hair".

put a "10" in the intersection of the circles.
put a "23" in the rectangle but not in either circle
put a 25-10 = "15" in the brown eyes circle that is not in the intersection
put a 15-10 = "5" in the blonde hair circle that is not in the intersection

Count up all the "" numbers to get 53
That is the number of people interviews.

Follow this same pattern to work the other problem you posted which involves three intersecting circles.

27. *If you want to make your Venn diagram on the computer, here are some symbols/shapes you might need:*

3 intersecting circles __ r, l, b

r-l-b __ 7
l-b __ 19-7=12
r-b __ 17-7=10
r-l __ 15-7=8
r __ 34-8-10-7=9
l __ 30-8-12-7=3
b __ 37-10-12-7=8

a) r=9
b) r + l + b =9+3+8=20
c) 7+12+10+8+9+3+8=57
d) l-b + r-b + r-l = 12+10+8 =30
e) 75-57=18

Chapter 2
Languages

Answer for Exercises (Page 45, Theory of Computer Science: Definitions and Examples. 3rd Ed.)

Part A Multiple Choices Questions

1. B
2. D
3. B
4. B
5. C
6. D
7. C
8. A
9. D
10. C
11. D
12. C
13. D
14. C
15. A
16. C
17. D
18. A
19. B
20. D
21. C
22. D
23. D
24. B
25. C
26. D
27. D
28. C
29. D
30. A
31. D
32. D
33. D
34. D
35. A
36. C
37. D
38. C
39. A
40. C
41. D
42. D
43. D
44. C
45. A
46. B
47. A
48. C
49. A
50. B

Part B Structured Questions

1.
   a) {λ}
   b) {0, 00, 000}
   c) {0, 00, 000, 0000, 00000, 000000}
   d) {01111, 001111, 0001111}
   e) {011011, 0110011, 01100011, 0011011, 00110011, 001100011, 00011011, 000110011, 0001100011}
   f) {111111}
   g) {λ, 0, 00, 000, 0000, 00000, 000000, . . .}
   h) {00, 000, 0000, 011, 00000, 0011, 000000, 00011, 110, 1100, 11000, 1111}

2. First we observe that L1 = {ε, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb}.
   a) L3 = {ε, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa}
   b) L4 = {aa, aaa}
   c) L5 = every way of selecting one element from L1 followed by one element from L4:
   {εaa, aaa, baa, aaaa, abaa, baaa, bbaa, aaaaa, aabaa, abaaa, abbaa, baaaa, babaa, bbaaa, bbbaa} ∪ {εaaa, aaaa, baaa, aaaaa, abaaa, baaaa, bbaaa, aaaaaa, aabaaa, abaaaa, abbaaa, baaaaa, babaaa, bbaaaa, bbbaaa}.
   Note that we've written εaa, just to make it clear how this string was derived. It should actually be written as just *aa*. Also note that some elements are in both of these sets (i.e., there's more than one way to derive them). Eliminating duplicates (since L is a set and thus does not contain duplicates), we get:
   {aa, aaa, baa, aaaa, abaa, baaa, bbaa, aaaaa, aabaa, abaaa, abbaa, baaaa, babaa, bbaaa, bbbaa, aaaaaa, aabaaa, abaaaa, abbaaa, baaaaa, babaaa, bbaaaa, bbbaaa}
   d) L6 = every string that is in L1 but not in L2: {ε, a, b, ab, ba, bb, aab, aba, abb, baa, bab, bba, bbb}.

3.
   a) Yes. n = 0 and m = 0.
   b) Yes. n = 1 and m = 0.
   c) Yes. n = 0 and m = 1.
   d) No. There must be equal numbers of a's and b's.
   e) Yes. n = 2 and m = 2.
   f) No. There must be equal numbers of a's and b's.

4. Y = {would, wood, could, cood}

5.
   a) {aa, bb, aab, bbb aaab, bbab}
   b) {aa, bb, baa, bbb, abaa, abbb}
   c) 8
   d) {*λ,* b, ab, bab, abb}
   e) {aa, bb, aab, bbb, aaab, bbab, aabb, bbbb}

6.

a) $X^0Y = Y = \{b\}$
$X^1Y = \{ab\}$
$X^2Y = \{aab\}$
$X^3Y = \{aaab\}$

b) $XY^0 = \{a\}$
$XY^1 = \{ab\}$
$XY^2 = \{abb\}$
$XY^3 = \{abbb\}$

c) $(XY)^0 = \{\lambda\}$
$(XY)^1 = \{ab\}$
$(XY)^2 = \{abab\}$
$(XY)^3 = \{ababab\}$

7. { λ , 0,1,00,01,10,11, ...}

8.

a) 2
b) 2
c) All strings over {*a*, b} that contain multiple of *a*

9.

Table 1

Substring	Prefix	Suffix	None
bba abab abbaa bba abb babb bab abbaaababb	abbaa abb abbaaababb	abb babb abbaaababb	baba bbb

10.

a) No
b) {*a*, *ab*, *ba*, *aa*, *aab, aba*, *abab*, *abba, baa, baab, baba*}

11.

a) No
b) No
c) No

12.

a) The language consisting of any concatenation of *a*'s and *b*'s of even length.

b) If S contains all possible strings of *a* and *b* of length 3, then all the words in S* will have length divisible by 3 and will include any concatenation of *a*'s and *b*'s (because S did).
By the product rule, there are 2*2*2 = 8 possible words of length 3:
S = {aaa, aab, aba, baa, abb, bba, bab, bbb}

c) S* = {λ and all possible strings of *a*'s and *b*'s of any length}.
This is because a ∈ S and b ∈ S, and any combination of these letters can result in any word.
Furthermore, an odd word + and odd word = an even word.
an odd word + an odd word + an odd word = an odd word.
So both odd and even words are included.

13.

a) Y = {00, 01,10, 11}
b) {10000, 10001, 10010, 10011}
c) {00100, 01100, 10100, 11100}
d) {10000100, 10001100, 10010100, 11100}

14.

a) 11(1 ∪ 0)*00
b) (11(1 ∪ 0)*) ∪ ((1 ∪ 0)*00)
c) 11(1 ∪ 0)*00

15. {*ab*, *aaa*, *abb, abaa*, *bab*, *baaa*}

16.

Table 2.

IN the language	**Describe the language in your own words**
aba, abba, bba, aabba, ababba, aaba	All strings over {*a*, *b*} that end with *ba*
a, ab, aab, ababab	All strings over {*a*, *b*} that do not contain substring *bb*
ab, a, ababab, aaaa,	All strings over {*a*, *b*} that do not contain substring *bb*
aaba, aabb, aaaba, ababb, baba, babb, ababa, ababb,	All strings over {*a*, *b*} that contain substring *ab* and end with *a* or *b*

17.

Table 3.

IN the language	**NOT IN the language**
a, aaaa, aaab, aaba, abaa, abab, abba, baaa, baab, baba	*b, bb, abb, aabb, abbb, babb, bbaa, bbab, bbba, bbbb*

18.

a) (R ∪ ϕ) is R since ϕ does not add anything to their union, and (Rλ) means appending nothing to all strings in R which is also R, hence they are the same.
b) (R ∪ λ) represents the set R and empty string.
c) (Rϕ) represents R.

19.

a) All string that contain only b's are not in the language. I.e.: *b*, *bb*, *bbb*, …

b) {*a, aa, ab, ba, aaa, aab, aba, abb*, *baa, bba, bab, aaaa, aaab, aaba, aabb, abaa, abab, baaa, baab, baba, babb, bbaa, bbab*}.

To understand how they are generated, we need first to list down all of them starting from the most basic factor as follows:

$(a + b)^0a(a + b)^0 = a$

$(a + b)^0a(a + b)^1 = aa, ab$

$(a + b)^1a(a + b)^0 = aa, ba$

$(a + b)^1a(a + b)^1 = aaa, aab, baa, bab$

$(a + b)^0a(a + b)^2 = a(a + b)(a + b) = aaa, aab, aba, abb$

$(a + b)^2a(a + b)^0 = (a + b)(a + b)a = aaa, aba, baa, bba$

$(a + b)^1a(a + b)^2 = (a + b)a(a + b)(a + b) = aaaa, aaab, aaba, aabb, baaa, baab, baba, babb$

$(a + b)^2a(a + b)^1 = (a + b)(a + b)a(a + b) = aaaa, aaab, abaa, abab, baaa, baab, bbaa, bbab$

Other than these factors are excluded because they will produce the string with more than 4 letters.

Now we will remove any duplicate strings and we get = {*a, aa, ab, ba, aaa, aab, aba, abb*, *baa, bba, bab, aaaa, aaab, aaba, aabb, abaa, abab, baaa, baab, baba, babb, bbaa, bbab*}.

20.

a) 001, 011, 0001, 0011; any string of length 3 or greater that is one or more 0's are followed by one or more 1's.

b) 111, 0111, 01011, 010101; any string that has at least three 1's.

c) 0, 1, 01, 0101; any string that has no substring 110.

d) 1, 01, 1011, 10110; any string that has no substring 00 after first 11 (if there is any 11 in the string).

21.

a) $a(a + b)^0$ = $a\lambda$ = a

$a(a + b)^1$ = $a(a + b)$ = aa, ab

$a(a + b)^2$ = $a(a + b)(a + b)$ = aaa, aab, aba, abb

$a(a + b)^3$ = $a(a + b)(a + b)(a + b)$ = aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb

So $a(a + b)^*$ = {a, aa, ab, aaa, aab, aba, abb, aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, ...}

b) a^0b^0 = λ

a^1b^0 = $a\lambda$

a^0b^1 = λb

a^1b^1 = ab

a^2b^1 = aab

a^1b^2 = abb

a^2b^2 = aabb

So a^*b^* = {λ, a, b, ab, aab, abb, aabb...}

c) $(ab)^0$ = λ
$(ab)^1$ = ab
$(ab)^2 = (ab)(ab)$ = abab
$(ab)^3 = (ab)(ab)(ab)$ = ababab
So (ab)* = {λ, ab, abab, ababab, ….}

22. Let X = abb; and Y = aaba
$(XY)^r = (abbaaba)^r$ = abaabba
$Y^rX^r. = (aaba)^r (abb)^r = (abaa)(bba)$ = abaabba

23. Length is n+1

24.

Three possible generated strings	Shortest string which is NOT in the language
a, ab, abb	λ, b
b, ab, aabb	λ, a
λ, a, bb, ababa	b
λ, ba, baba, bababa	a, b
ab, aaab, abbb	λ, a, b
a, ab, abab, aab, aba	b
ab, aa, ba, bb	λ, a, b

25. 0*11*222*

26. {*a*, *ba*}*

27. a*ba*b*b^+a* or (a + b)*b(a + b)*b(a + b)* or a*ba*b(a + b)*

28.
a) the language of all strings over {*a*, *b*} that starts and ends with *aa*.
b) the language of all strings over {*a*, *b*} with *b* as the second letter.

29.
a) $(a + b)^+$
b) $(aaa)^+$
c) $(aa)^+b(bb)^*(aa)^+$
d) $a(aa)^* + bb(bb)^*$
e) $(a + b)^*(aa + bb)(a + b)^*$
f) $aa(a + b)^* + (a + b)^*aa$
g) $aa(a + b)^*aa$

30.
a) $a^*b^*c^*$.
b) $a^+b^*c^* \cup a^*b^+c^* \cup a^*b^*c^+$.

31.

a) *aaabb* and *a* are members, *aba* and *ba* are not members.
b) *aa* and *abababaa* are members, *a* and *ababa* are not members.

32.

a) $a^*b^*c^*$.
b) $a^+b^*c^* \cup a^*b^+c^* \cup a^*b^*c^+$.
c) $(a \cup b \cup c)(a \cup b \cup c)(a \cup b \cup c)$.
d) $(a \cup b \cup c)^0 \cup (a \cup b \cup c)^1 \cup (a \cup b \cup c)^2$.
e) $(a \cup b \cup c)^+(a \cup b \cup c)^+(a \cup b \cup c)^+$.
f) $(a \cup b)^+ab(a \cup b)^* \cup (a \cup b)^*ab(a \cup b)^+$
g) a^+b^+
h) $(a \cup b)^*aa(a \cup b)^*bb(a \cup b)^* \cup (a \cup b)^*bb(a \cup b)^*aa(a \cup b)^*$. The two expressions joined by the $\cup$ indicate that the *aa* may precede or follow the *bb*.
i) $(a \cup b)^*aa(a \cup b)^*aa(a \cup b)^*$
j) The leading *a* and trailing *cc* explicitly placed in the expression $a(a \cup c)^*b(a \cup c)\ ^*b(a \cup c)^*cc$. Any number of *a*'s and *c*'s, represented by the expression $(a \cup c)^*$ may surround the two *b*'s.
k) At first glance it may seem that the desired language is given by the regular expression
$$(a \cup b)^*ab(a \cup b)^*ba(a \cup b)^* \cup (a \cup b)^*ab(a \cup b)^*ba(a \cup b)^*.$$
Clearly every string in this language has substrings *ab* and *ba*. Every string described by the preceding expression has length four or more. Have we missed some strings with the desired property? Consider *aba*, *bab*, and *aaaba*. A regular expression for the set of strings containing substrings *ab* and *ba* can be obtained by adding
$(a \cup b)^*aba(a \cup b)^* \cup (a \cup b)^*bab(a \cup b)^*$ to the expression above.
l) $(a \cup bc^+ \cup c)^*$
m) The language consisting of strings over {a, b} containing exactly three *a*'s is defined by the regular expression b*ab*ab*ab*. Applying the Kleene star to the preceding expression produces strings in which the number of a's is a multiple of three. However, (b*ab*ab*ab*)* does not contain strings consisting of b's alone; the null string is the only string that does have at least three a's. To obtain strings with no a's, b* is concatenated to the end of the preceding expression. Strings consisting solely of b's are obtained from (b*ab*ab*ab*)*b* by concatenating, from (b*ab*ab*ab*)* with b's from b*. The regular expression (b*ab*ab*a)b* defines the same language.
n) $a^*(b \cup c)a^*(b \cup c)a^*(b \cup c)a^*$ or $a^*((b \cup c)a^*)^3$
o) In the regular expression
$$(ab \cup ba \cup aba \cup b)^*,$$
a's followed by a *b* are produced by the first component, a's preceded by a *b* by the second component, and two *a*'s "covered" by a single *b* by the third component. The inclusion of *b* within the scope of the * operator allows any number additional *b*'s to occur anywhere in the string.
Another regular expression for this language is $((a \cup \lambda)b(a \cup \lambda))^*$.
p) (b + ab + abb)* or (ab)* + a* or $(b \cup ab)^*(a \cup \lambda)$
q) (b + ab + aab)(a + b)*
r) Every *aa* in the expression (b ∪ ab ∪ aab)* is followed by at least one *b*, which precludes the occurrence of *aaa*. However, every string in this expression ends with *b*

while strings in the language may end in *a* or *aa* as well as *b*. Thus the expression

$$(b \cup ab \cup aab)^*(\lambda \cup a \cup aa)$$

represents the set of all strings over {a, b} that do not contain the substring *aaa*.

s) A string can begin with any number of *b*'s. Once a's are generated, it is necessary to ensure that an occurrence of *ab* is either followed by another *b* or ends the string. The regular expression $b^*(a \cup abb^+)^*(\lambda \cup ab)$ describes the desired language.

t) $(bab \cup b)^*aa(bab \cup b)^*$

u) To obtain an expression for the strings over {a, b, c} with an odd number of occurrences of the substring *ab* we first construct an expression *w* defining strings over {a, b, c} that do not contain the substring *ab*,

$$w = b^*(a \cup cb^*)^*:$$

Using *w*, the desired set can be written $(wabwab)^*wabw$.

v) $((b \cup c)(b \cup c))^*(ab \cup ac \cup ba \cup ca)((b \cup c)(b \cup c))^*$

w) $a(aa)^*b(aa)^*b(aa)^* \cup (aa)^*ba(aa)^*b(aa)^* \cup (aa)^*b(aa)^*ba(aa)^*$

x) The regular expression $(b^*ab^*a)^*b^* \cup (a^*ba^*b)^*a^*ba^*$ defines strings over {a, b} with an even number of *a*'s or an odd number of *b*'s. This expression is obtained by combining an expression for each of the component subsets with the union operation.

y) $([aa \cup ab(bb)^*ba] \cup [(b \cup a(bb)^*ba)(a(bb)^*a)^*(b \cup ab(bb)^*a)])^*$.

33.

Table 1. Table 2.

No.	Matching no. of language	Generate 3 possible strings
1.	9	*aa, b, aba, abab*
2.	11	*aa, aab, baa, aaa*
3.	12	*bb, abb, aabb, aaaba*
4.	1	*ba, ab, bba, abb*
5.	2	*aa, ab, aaa, aab*
6.	6	*ba, aba, abbba*
9.	3	*a, bb, abb, ba*
10.	14	*bb, abb, bba, abba*
11.	13	*bb, abb, bba, bbb*
12.	4	*aaa, baaa, aaab*
13.	5	*b, aaa, aaaaaa, bb*
14	15	*aba, abba, aaba*
15	10	*a, ba, ab*

Answer for Exercises (Page 45, Theory of Computer Science: Definitions and Examples. 3rd Ed.)

Part A Multiple Choices Questions

1. B
2. B
3. D
4. C
5. D
6. D
7. D
8. C
9. D
10. A
11. A
12. C
13. C
14. B
15. C
16. B
17. A
18. A
19. D
20. D
21. B
22. D
23. B
24. C
25. C
26. A
27. D
28. A
29. D
30. C
31. B
32. D
33. C
34. A
35. B
36. B
37. B
38. A
39. A
40. A
41. A
42. B
43. C
44. C
45. D
46. D

Part B Structured Questions

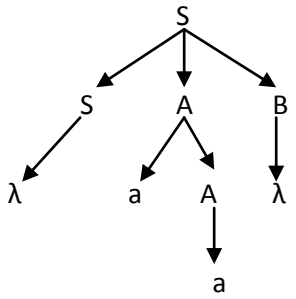
1.

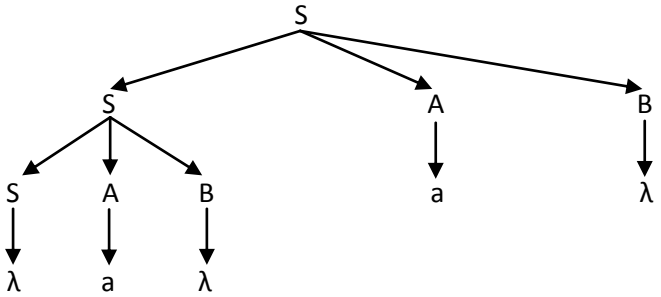
a) $S \Rightarrow SAB$
$\Rightarrow SABAB$
$\Rightarrow \lambda ABAB$
$\Rightarrow \lambda aBAB$
$\Rightarrow \lambda abBAB$
$\Rightarrow \lambda abb BAB$
$\Rightarrow \lambda abb \lambda AB$
$\Rightarrow \lambda abb \lambda aAB$
$\Rightarrow \lambda abb \lambda aaB$
$\Rightarrow \lambda abb \lambda aabB$
$\Rightarrow \lambda abb \lambda aab \lambda$
$\Rightarrow abbaab$

b) $S \Rightarrow SAB$
$\Rightarrow \lambda AB$
$\Rightarrow \lambda aAB$
$\Rightarrow \lambda aaB$
$\Rightarrow \lambda aa \lambda$
$\Rightarrow aa$

$S \Rightarrow SAB$
$\Rightarrow SABAB$
$\Rightarrow \lambda ABAB$
$\Rightarrow \lambda aBAB$
$\Rightarrow \lambda a \lambda AB$
$\Rightarrow \lambda a \lambda aB$
$\Rightarrow \lambda a \lambda a \lambda$
$\Rightarrow aa$

c)





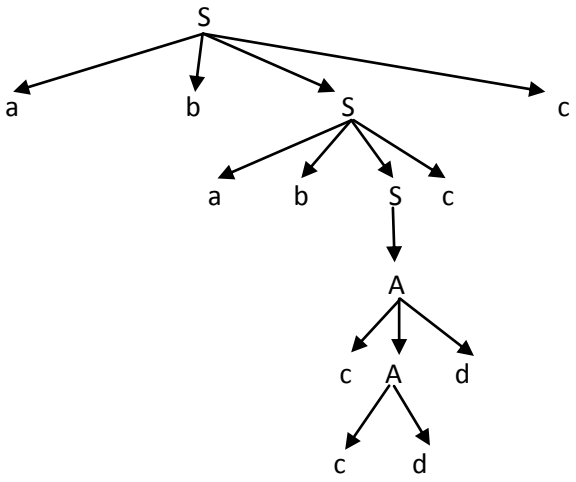
d) (a*ab*)*

2.

b)

Derivation	**Rule**
$S \Rightarrow abSc$	$S \rightarrow abSc$
$\Rightarrow ababScc$	$S \rightarrow abSc$
$\Rightarrow ababAcc$	$S \rightarrow A$
$\Rightarrow ababcAdcc$	$A \rightarrow cAd$
$\Rightarrow ababccddcc$	$A \rightarrow cd$

c) The derivation tree corresponding to the preceding derivation is



d) $L(G) = \{(ab)^n c^m d^m c^n \mid n \geq 0; m > 0\}$

3.

a)

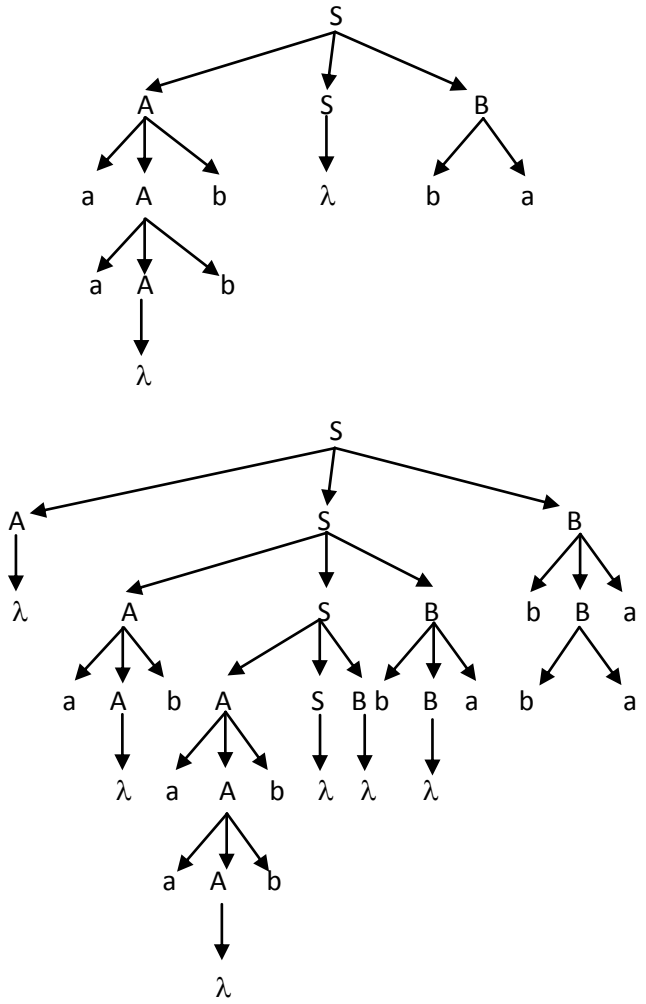
Derivation	**Rule**
$S \Rightarrow ASB$	$S \rightarrow ASB$
$\Rightarrow aAbSB$	$A \rightarrow aAb$
$\Rightarrow aaAbbSB$	$A \rightarrow aAb$
$\Rightarrow aa\lambda bbSB$	$A \rightarrow \lambda$
$\Rightarrow aa\lambda bb\lambda B$	$S \rightarrow \lambda$
$\Rightarrow aa\lambda bb\lambda ba$	$B \rightarrow ba$
$\Rightarrow aabbba$	

b)

Derivation	**Rule**
$S \Rightarrow ASB$	$S \rightarrow ASB$
$\Rightarrow ASbBa$	$B \rightarrow bBa$
$\Rightarrow ASbbaa$	$B \rightarrow ba$
$\Rightarrow AASBbbaa$	$S \rightarrow ASB$
$\Rightarrow AASbBabbaa$	$B \rightarrow bBa$
$\Rightarrow AASb\lambda abbaa$	$B \rightarrow \lambda$
$\Rightarrow AAASBb\lambda abbaa$	$S \rightarrow ASB$
$\Rightarrow AAAS\lambda b\lambda abbaa$	$B \rightarrow \lambda$
$\Rightarrow AAA\lambda\lambda b\lambda abbaa$	$S \rightarrow \lambda$

$\Rightarrow AAaAb\lambda\lambda b\lambda abbaa \quad A \rightarrow aAb$
$\Rightarrow AAaaAbb\lambda\lambda b\lambda abbaa \quad A \rightarrow aAb$
$\Rightarrow AAaa\lambda bb\lambda\lambda b\lambda abbaa \quad A \rightarrow \lambda$
$\Rightarrow AaAbaa\lambda bb\lambda\lambda b\lambda abbaa \quad A \rightarrow aAb$
$\Rightarrow Aa\lambda baa\lambda bb\lambda\lambda b\lambda abbaa \quad A \rightarrow \lambda$
$\Rightarrow \lambda a\lambda baa\lambda bb\lambda\lambda b\lambda abbaa \quad A \rightarrow \lambda$
$\Rightarrow abaabbbabbaa$

c)



d) $L(G) = \{a^{n1}b^{n1} \ldots a^{nk}b^{nk}\, b^{m1}a^{m1} \ldots b^{ml}a^{ml} \mid n_i, m_i > 0;\ k, l \geq 0;\ k \leq l\}$

4.

a) $S \Rightarrow AB$
$\Rightarrow aAB$
$\Rightarrow aaB$
$\Rightarrow aaAB$
$\Rightarrow aaaB$
$\Rightarrow aaab$

b) S $\Rightarrow$ AB
$\Rightarrow$ AAB
$\Rightarrow$ AaB
$\Rightarrow$ Aab
$\Rightarrow$ aAab
$\Rightarrow$ aaab

5.

Set notations	**Descriptions**	**Shortest string and 3 other strings**
$\{a^{2i}b^ib^*, i \geq 0\}$ or $\{(aa)^ib^j \mid i > 0, j \geq i\} \cup \{\lambda\}$ or $\{a^{2i}b^j \mid i > 0, j \geq i\} \cup \{\lambda\}$.	The S rules S $\rightarrow$ *aaSB* and S $\rightarrow \lambda$ generate an equal number of leading *aa's* and trailing *B*'s. Each *B* is transformed into one or more *b*'s. Since at least one application of the rule S $\rightarrow$ *aaSB* is necessary to generate *b*'s, strings consisting solely of *b*'s are not in the language. The language of the grammar is $\{(aa)^ib^j \mid i > 0, j \geq i\} \cup \{\lambda\}$ or we can write $\{a^{2i}b^j \mid i > 0, j \geq i\} \cup \{\lambda\}$.	aab aabb, aaaabb, aaaabbb
$\{a^nc^+b^{2n}, n \geq 0\}$ or $\{a^nc^m(bb)^n \mid n \geq 0, m > 0\}$ or $\{a^nc^mb^{2n} \mid n \geq 0, m > 0\}$.	The recursion of S rules *S* $\rightarrow$ *aSbb* generate an equal number of leading *a's* and trailing *bb*'s. The recursion is terminated by an application of the rule S $\rightarrow$ A. Each *A* is transformed into one or more *c*'s. The language of the grammar is $\{a^nc^m(bb)^n \mid n \geq 0, m > 0\}$ or we can write $\{a^nc^mb^{2n} \mid n \geq 0, m > 0\}$.	c aacbbbb accbb acccbb
$\{(ab)^n(cd)^m(ba)^m(dc)^n, n \geq 1, m \geq 0\}$	Repeated applications of the recursive rule S $\rightarrow$ *abSdc* produce a string of the form $(ab)^iS(dc)^i$, $i \geq 0$. The recursion is terminated by an application of the rule S $\rightarrow$ A. The A rules produce strings of the form $(cd)^j(ba)^j$, $j \geq 0$. The language generated by the rules is $\{(ab)^i(cd)^j(ba)^j(dc)^i \mid i \geq 0, j \geq 0\}$.	abdc / cdba abcdbadc, abcdcdbabadc, ababcdcdbabadcdc
$\{a^nc^ma^pb^pd^mb^n, m \geq 1, n \geq 0, p \geq 1\}$	Repeated applications of the recursive rule S $\rightarrow$ *aSb* produce a string of the form a^mSb^m, $m \geq 0$. The recursion is terminated by an application of the rule S $\rightarrow$ A. The A rules produce strings of the form c^nAd^n, $n > 0$. While the B rules produce strings of the form a^iBb^i, $i > 0$. The language generated by the rules is $\{a^mc^na^ib^id^nb^m \mid m \geq 0; n, i > 0\}$	cabd aacccabdddbb, acaabbdb, aacaabbdbb
$\{a^kb^j \mid 0 < k \geq j \geq 2k\}$	The rule S $\rightarrow$ *aSB* | *ab* generates strings of the form a^kSB^k, by applying the first option of the rewrite rule k times, or a^kabB^k by applying the second option to replace S. The rule B $\rightarrow$ b | bb, applied *k* times to the strings generated above will eventually yield $a^kab(b \cup bb)^k$. This can be written as $L = \{a^kb^j \mid 0 < k \geq j \geq 2k\}$.	ab abb, abbb, aabb

6. $S \rightarrow AB$
   $A \rightarrow aAbb \mid abb$
   $B \rightarrow cB \mid c$

7. The language consisting of the set of strings $\{a^n b^m c^{2n+m} \mid m, n > 0\}$ is generated by
   $S \rightarrow aScc \mid aAcc$
   $A \rightarrow bAc \mid bc$
   For each leading a generated by the *S* rules, two *c*'s are produced at the end of the string. The rule *A* rules generate an equal number of *b*'s and *c*'s.

8. This grammar can be constructed by first generating as many *a*'s as *c*'s to the left and right. Subsequently generate as many *b*'s as *c*'s in the middle. This gives us as many *a*'s and *b*'s as *c*'s. Then we create possibly more *c*'s.
   S → aSc | B Generate $a^n Bc^n$, $n \geq 0$ by applying left option *n* times.
   B → bBc | C Use $a^n Bc^n$ and this rule to generate $a^n b^m Cc^{n+m}$, $n, m \geq 0$.
   C → cC | λ Use $a^n b^m Cc^{n+m}$ and this rule to get $a^n b^m c^j c^{n+m}$, $n, m, j \geq 0\}$.
   This gives us the language L = $\{a^n b^m c^j c^{n+m} \mid n, m, j \geq 0\}$. Let $i = j + n + m$ to get: $\{a^n b^m c^i \mid 0 \leq n + m \leq i\}$.

9. S → AAaSb|λ or S → ASb|λ
   A→ a | λ A→ a | aa | aaa

10. S → LR
    L → aLb | λ
    R → bLa | λ

11. The grammar of this language is
    S → SaSbS | SbSaS | λ

12. An *a* can be specifically placed in the middle position of a string using the rules
    A → aAa | aAb | bAa | bAb | a
    The termination of the application of recursive A rules by the rule A → a inserts the symbol *a* into the middle of the string. Using this strategy, the grammar
    S → aAa | aAb | bBa | bBb | a | b
    A → aAa | aAb | bAa | bAb | a
    B → aBa | aBb | bBa | bBb | b
    generates all strings over fa; bg with the same symbol in the beginning and middle positions. If the derivation begins with an *S* rule that begins with an *a*, the A rules ensure that an a occurs in the middle of the string. Similarly, the *S* and *B* rules combine to produce strings with a b in the first and middle positions.

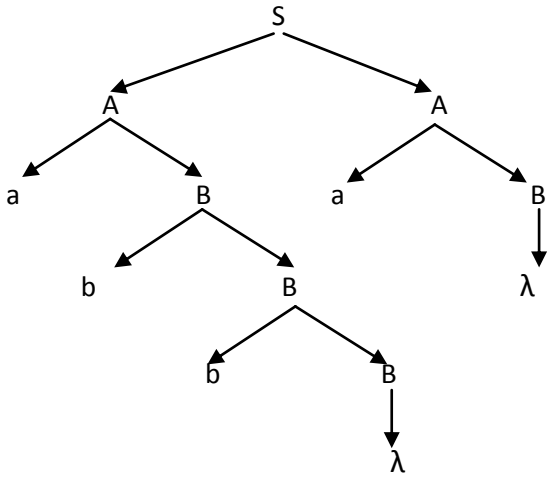
13.
    a) $(a + b)^* aa(a + b)^*$
    b) S → *AaaA* A → aA | bA | λ

14.

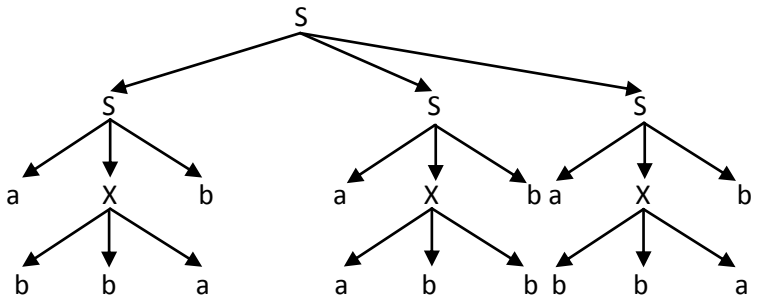
a) a^*bb

b) $S \rightarrow Abb \qquad A \rightarrow aA \mid \lambda$

15. $S \Rightarrow AA \Rightarrow aBA \Rightarrow abBA \Rightarrow abbBA \Rightarrow abb\lambda A \Rightarrow abb\lambda aB \Rightarrow abb\lambda a\lambda \Rightarrow abba$



16. $S \Rightarrow SSS \Rightarrow aXbSS \Rightarrow abbabSS \Rightarrow abbabaXbS \Rightarrow abbabaabbbS \Rightarrow abbabaabbbaXb \Rightarrow abbabaabbbabbab$



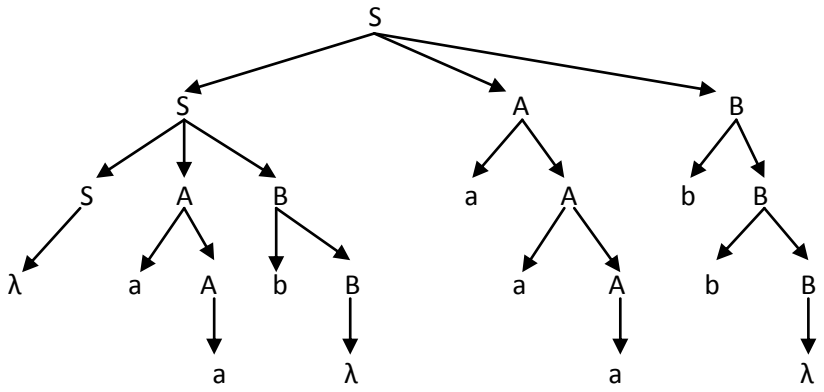
17. $S \rightarrow U111 \qquad U \rightarrow 0U \mid 1U \mid \lambda$

18.

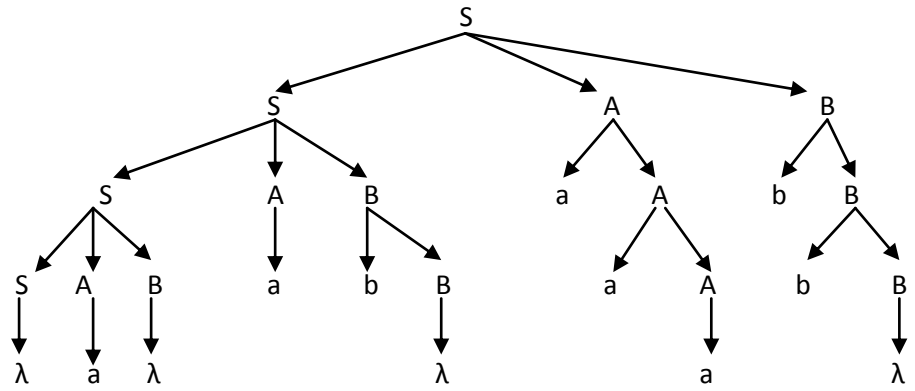
a) $S \Rightarrow SAB$
$\Rightarrow SABAB$
$\Rightarrow \lambda ABAB$
$\Rightarrow \lambda aABAB$
$\Rightarrow \lambda aaBAB$
$\Rightarrow \lambda aabBAB$
$\Rightarrow \lambda aab\lambda AB$
$\Rightarrow \lambda aab\lambda aAB$
$\Rightarrow \lambda aab\lambda aaAB$
$\Rightarrow \lambda aab\lambda aaaB$
$\Rightarrow \lambda aab\lambda aaabB$

⇒ λaabλaaabbB
⇒ λaabλaaabbλ
⇒ aabaaabb

b)



c)



d) (a*ab*)*

19.

a) Analysis: the language will produce strings with exactly 3 *a*'s and end with *a*. in between of the *a*'s can have as many *b*'s as we like. The pattern of the string will be : BaBaBa.

S → bS | aA
A → bA | aB
B → bB | a

b) Analysis: the language will produce strings with at least one a and one b. The pattern of the string will be : AbBBaA.

S → aS | bA
A → bbA | aB
B → aB | λ

20.

a) S → aA
A → aA | aB | λ

$B \rightarrow bA$

b) $S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaaaB \Rightarrow aaaabA \Rightarrow aaaabaB \Rightarrow aaaababA \Rightarrow aaaababaA \Rightarrow aaaababa$.

21. $L(G_1) = \{b^m a^n b^m : n \geq 0, m \geq 0\}$

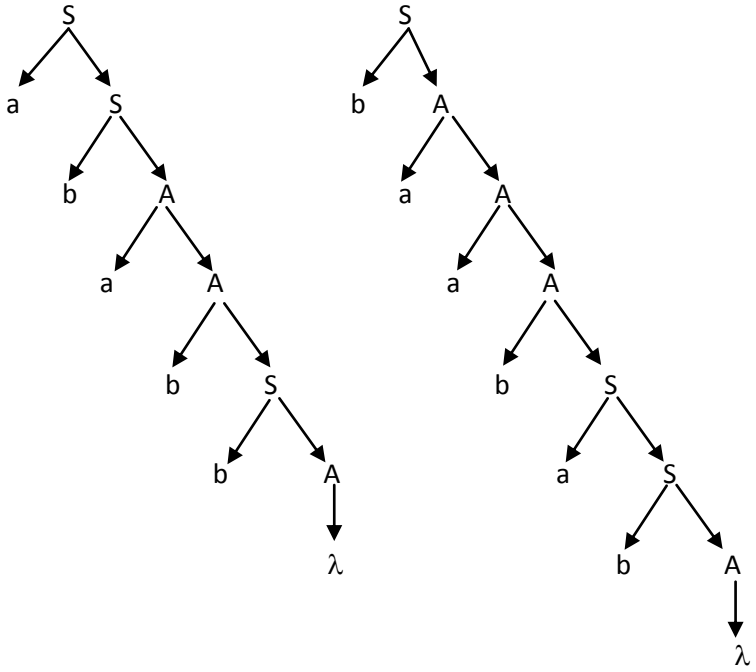
22. G1 = ({a, b, S}, {a, b}, S, P) with P defined by: $S \rightarrow aSbb \mid \lambda$

23. a) $S \Rightarrow bA \Rightarrow bbS \Rightarrow bbaS \Rightarrow failed.$

$S \Rightarrow aS \Rightarrow abA \Rightarrow abaA \Rightarrow ababS \Rightarrow ababbA \Rightarrow ababb\lambda$

$S \Rightarrow bA \Rightarrow baA \Rightarrow baaA \Rightarrow baabS \Rightarrow baabaS \Rightarrow baababA \Rightarrow baabab\lambda$

b)



24.

a) $a(a + b)^*b$

b) aa^*bb^* or a^+b^+

c) a^*bba^*

d) $(a^* + ba^*b)^*$

25. $S \rightarrow aS \mid aA \mid \lambda$

$A \rightarrow bA \mid bB \mid \lambda$

$B \rightarrow cB \mid \lambda$

26. The language $(a \cup b)^*aa(a \cup b)^*bb(a \cup b)^*$ is generated by

G_1: $S_1 \rightarrow aS_1 \mid bS_1 \mid aA$

$A \rightarrow aB$

$B \rightarrow aB \mid bB \mid bC$

$C \rightarrow bD$

$D \rightarrow aD \mid bD \mid \lambda$

G_2 generates the strings $(a \cup b)^*bb(a \cup b)^*aa(a \cup b)^*$

G_2: $S_2 \rightarrow aS_2 \mid bS_2 \mid bE$
$E \rightarrow bF$
$F \rightarrow aF \mid bF \mid aG$
$G \rightarrow aH$
$H \rightarrow aH \mid bH \mid \lambda$

A grammar G that generates

$(a \cup b)^*aa(a \cup b)^*bb(a \cup b)^* \cup (a \cup b)^*bb(a \cup b)^*aa(a \cup b)^*$

can be obtained from G_1 and G_2. The rules of G consist of the rules of G1 and G2 augmented with $S \rightarrow S_1 \mid S_2$ where S is the start symbol of the composite grammar. The alternative in these productions corresponds to the $\cup$ in the definition of the language.

While the grammar described above generates the desired language, it is not regular. The rules $S \rightarrow S_1 \mid S_2$ do not have the form required for rules of a regular grammar. A regular grammar can be obtained by explicitly replacing S_1 and S_2 in the S rules with the right-hand sides of the S_1 and S_2 rules. The S rules of the new grammar are

$S \rightarrow aS1 \mid bS1 \mid aA$
$S \rightarrow aS2 \mid bS2 \mid bE$

The strategy used to modify the rules $S \rightarrow S_1 \mid S_2$ is an instance of a more general rule modification technique known as removing chain rules.

27. $S \rightarrow aS \mid bS \mid aA$
$A \rightarrow aB$
$B \rightarrow aB \mid bB \mid aC$
$C \rightarrow aD$
$D \rightarrow aD \mid bD \mid \lambda$

28. G_1 generates the strings $(a \cup b)^*ab(a \cup b)^*ba(a \cup b)^*$

G_1: $S_1 \rightarrow aS_1 \mid bS_1 \mid aA$
$A \rightarrow bB$
$B \rightarrow aB \mid bB \mid bC$
$C \rightarrow aD$
$D \rightarrow aD \mid bD \mid \lambda$

G_2 generates the strings $(a \cup b)^*ba(a \cup b)^*ab(a \cup b)^*$

G_2: $S_2 \rightarrow aS_2 \mid bS_2 \mid bE$
$E \rightarrow aF$
$F \rightarrow aF \mid bF \mid aG$
$G \rightarrow bH$
$H \rightarrow aH \mid bH \mid \lambda$

Thus a grammar G that generates $(a \cup b)^*ab(a \cup b)^*ba(a \cup b)^* \cup (a \cup b)^*ba(a \cup b)^*ab(a \cup b)^*$

$S \rightarrow aS1 \mid bS1 \mid aA$
$S \rightarrow aS2 \mid bS2 \mid bE$
$S_1 \rightarrow aS_1 \mid bS_1 \mid aA$
$A \rightarrow bB$
$B \rightarrow aB \mid bB \mid bC$
$C \rightarrow aD$

$D \rightarrow aD \mid bD \mid \lambda$
$S_2 \rightarrow aS_2 \mid bS_2 \mid bE$
$E \rightarrow aF$
$F \rightarrow aF \mid bF \mid aG$
$G \rightarrow bH$
$H \rightarrow aH \mid bH \mid \lambda$ or

G: $S \rightarrow aS \mid bS \mid aA \mid bE$
$A \rightarrow bB$
$B \rightarrow aB \mid bB \mid bC$
$C \rightarrow aD$
$D \rightarrow aD \mid bD \mid \lambda$
$E \rightarrow aF$
$F \rightarrow aF \mid bF \mid aG$
$G \rightarrow bH$
$H \rightarrow aH \mid bH \mid \lambda$

29. $S \rightarrow bS \mid aA \mid \lambda$
$A \rightarrow bA \mid aB$
$B \rightarrow bB \mid aS$

30. The language $((a \cup \lambda)b(a \cup \lambda))^*$ is generated by the grammar

$S \rightarrow aA \mid bB \mid \lambda$
$A \rightarrow bB$
$B \rightarrow aS \mid bB \mid \lambda$

This language consists of all strings over {*a*, *b*} in which every *a* is preceded or followed by a *b*. An *a* generated by the rule $S \rightarrow aA$ is followed by a *b*. An *a* generated by $B \rightarrow aS$ is preceded by a *b*.

31. When we start and we read a *b* then we are no closer to reading *aba* and we remain in the same state. However, if we read an *a*, we may encounter *ba* next, so we proceed to a state *A*, where we may not read *ba* next. If we read an *a* in this state *A* there still might follow *Ba*, so we remain in state *A*. On the other hand, when we read a *B* in state *A*, we end up in a state where we may not read an *a* because we have just read *ab*. If the character we read then is a *b* we are back at the initial state where the string to follow may not be *aba*. Of course, we are always allowed to stop reading and accept that word in any state. This yields the following grammar

$S \rightarrow bS \mid aA \mid \lambda$ string to follow may not be *aba*
$A \rightarrow aA \mid bB \mid \lambda$ string to follow may not be *ba*
$B \rightarrow bS \mid \lambda$ string to follow may not be *a*

32. G: $S \rightarrow bS \mid aA \mid aC$
$A \rightarrow aB$
$B \rightarrow bB \mid bD \mid \lambda$
$C \rightarrow bS$
$D \rightarrow aB$

33. G: $S \rightarrow aS \mid aA \mid bD \mid bG$
$A \rightarrow bB$
$B \rightarrow aB \mid bC \mid \lambda$
$C \rightarrow aC \mid \lambda$
$D \rightarrow aE$
$E \rightarrow aE \mid bF \mid \lambda$
$F \rightarrow aF \mid \lambda$
$G \rightarrow aG \mid bH$
$H \rightarrow aI \mid \lambda$
$I \rightarrow aI \mid \lambda$

34. The variables of the grammar indicate whether an even or odd number of *ab*'s has been generated and the progress toward the next *ab*. The interpretation of the variables is

Variable	Parity	Progress toward *ab*
S	even	none
A	even	a
B	odd	none
C	odd	a

The rules of the grammar are
$S \rightarrow aA \mid bS$
$A \rightarrow aA \mid bB$
$B \rightarrow aC \mid bB \mid \lambda$
$C \rightarrow aC \mid bS \mid \lambda$
A derivation may terminate with a λ-rule when a B or a C is present in the sentential form since this indicates that an odd number of *ab*'s have been generated.

35. The objective is to construct a grammar that generates the set of strings over {a, b} containing an even number of *a*'s or an odd number of *b*'s. In the grammar,
$S \rightarrow aOa \mid bEa \mid bOb \mid aEb \mid \lambda$
$Ea \rightarrow aOa \mid bEa \mid \lambda$
$Oa \rightarrow aEa \mid bOa$
$Ob \rightarrow aOb \mid bEb \mid \lambda$
$Eb \rightarrow aEb \mid bOb$
the Ea and Oa rules generate strings with an even number of a's. The derivation of a string with a positive even number of a's is initiated by the application of an S rule that generates either Ea or Oa. The derivation then alternates between occurrences of Ea and Oa until it is terminated with an application of the rule $Ea \rightarrow \lambda$.
In a like manner, the Ob and Eb rules generate strings with an odd number of b's. The string aab can be generated by two derivations; one beginning with the application of the rule $S \rightarrow aOa$ and the other beginning with $S \rightarrow aEb$.

36.

a) $\{a^n b^m b^n \mid n \geq 0, m > 0\}$ or we can write $\{a^m b^n \mid n > m \geq 0\}$.

b) $\{a^n b^{2m} d^{2m} a^{2n} \mid n \geq 0, m > 0\}$

c) $\{a^n a^m b^n \mid n, m \geq 0\}$

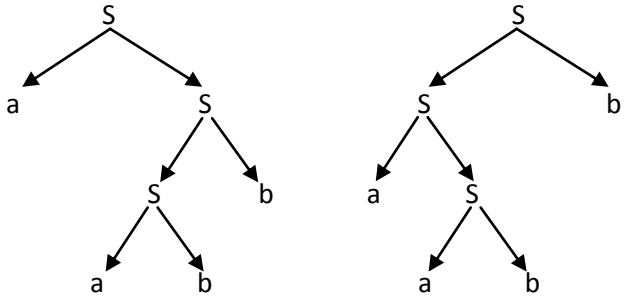
37.

a) $(a^+b^+)^+$

b) The rules $S \rightarrow aS$ and $S \rightarrow Sb$ allow the generation of leading *a*'s or trailing *b*'s in any order. Two leftmost derivations for the string *aabb* are

$S \Rightarrow aS$ $S \Rightarrow Sb$
$\Rightarrow aSb$ $\Rightarrow aSb$
$\Rightarrow aabb$ $\Rightarrow aabb$

c) The derivation trees corresponding to the derivations of (b) are



38. The regular expression for this language is $a^+b^+b \cup \lambda$.

39. $\{a^m a^n b^n \mid m > 0, n \geq 0\}$ or we can write $\{a^m b^n \mid m > n \geq 0\}$.

40.

a) The language {a, b}*
b) The set {a, b}*{a}
c) The set {ba}*{b}
d) The set {b*ab*}
e) {a}*{b}{a}*{b}{a}*, the set of strings in *{a, b} containing exactly two b's
f) The set of strings over {a, b} that are not palindromes but could be made into palindromes by changing one *a* to *b* or vice versa
g) The set of even-length strings in {a, b}*
h) The set of odd-length strings in {a, b}*

41.

a) $S \rightarrow aSb \mid Sb \mid \lambda$
b) $S \rightarrow aSb \mid Sb \mid b$
c) $S \rightarrow aSbb \mid \lambda$
d) $S \rightarrow aSb \mid aSbb \mid \lambda$
e) $S \rightarrow aSbb \mid aSb \mid aS \mid \lambda$
f) $S \rightarrow aAb \mid aA$; $A \rightarrow aAbb \mid aAb \mid aA \mid \lambda$

42.

Table 1.

No.	Language
1.	(a + b)*
2.	$(a + b)^+$
3.	(a + b)
4.	(ab)
5.	a*bb
6.	aa*bb*
7.	a*bba*
8.	a(a + b)*b
9.	(a + b)*bb(a + b)*
10.	(a + b)*aa(a + b)*

Table 2.

No.	Grammars	CFG or RG
6	S → aA A → aA | bB B → bB | λ	RG
5	S → Abb A → aA | λ	RG
2	S → aS | bS | a | b	RG
10	S → AaaA A → aA | bA | λ	CFG
4	S → aA A → b	RG
9	S → aS | bS | bA A → bB B → aB | bB | λ	RG
8	S → aA A → aA | bA | b	RG
1	S → aS | bS | λ	RG
7	S → aS | bA A → bB B → aB | λ	RG
3	S → a | b	RG

43.

Table 3.

No.	Language
1.	$L = \{ww^R : w \in (a + b)^*\}$
2.	$L = \{a^n b^{2m} c^n \mid n, m > 0\}$
3.	L = (bb*a + ab*b)*
4.	Let L be the language of all strings over {a, b} that have the same number of *a*'s as *b*'s
5.	$L = \{a^n b^n \mid n > 0\} \cup \{b^n a^n \mid n > 0\}$

Table 4.

No.	Grammars	CFG or RG	Generate 3 possible strings
4	S → aSb | bSa | SS | λ	**CFG**	*ab, ba, aaba*
3	S → bA | aB | λ A → bA | aS B → bB | bS	**RG**	
5	S → X | Y X → aXb | ab Y → bYa | ba	**CFG**	
2	S → ASC | B A → a C → c B → bbB | bb	**CFG**	
1	S → aSa | bSb | λ	**CFG**	

Chapter 4
Finite Automata

Answer for Exercises (Page 151, Theory of Computer Science: Definitions and Examples. 3rd Ed.)

Part A Multiple Choices Questions

1. B
2. B
3. C
4. A
5. B
6. D
7. D
8. B
9. A
10. A
11. D
12. D
13. B
14. B
15. B
16. B
17. D
18. B
19. C
20. C
21. C
22. A
23. D
24. C

Part B Structured Questions

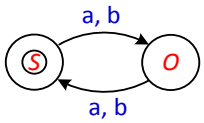
1. What is the meaning of *error* or *trap state*?
   Non-accepting state

2. What are the characteristics of NFA compared to DFA?

Features	**DFA**	**NFA**
Choice of transition	Single	Multiple
Input label	Alphabets only	Alphabets or λ or ε
Undefined transitions	No	Possible

3.

a)



DFA

b) $[q_0, \mathbf{abbb}] \vdash [q_1, bbb]$
$\vdash [q_0, bb]$
$\vdash [q_1, b]$
$\vdash [q_0, \lambda]$
Stop at q_0 (final state) and all alphabet is traced (string is empty);
Hence, string **abbb is accepted** by the machine.

$[q_0, \mathbf{bbb}] \vdash [q_1, bb]$
$\vdash [q_0, b]$
$\vdash [q_1, \lambda]$
All alphabet is traced (string is empty) but stop at q_1 (not final state) ;
Hence, string **bbb is not accepted** by the machine.

$[q_0, \mathbf{baa}] \vdash [q_1, aa]$
$\vdash [q_0, a]$
$\vdash [q_1, \lambda]$
All alphabet is traced (string is empty) but stop at q_1 (not final state) ;
Hence, string **baa is not accepted** by the machine.

$[q_0, \mathbf{baab}] \vdash [q_1, aab]$
$\vdash [q_0, ab]$
$\vdash [q_1, b]$
$\vdash [q_0, \lambda]$
Stop at q_0 (final state) and all alphabet is traced (string is empty);
Hence, string **baab is accepted** by the machine.

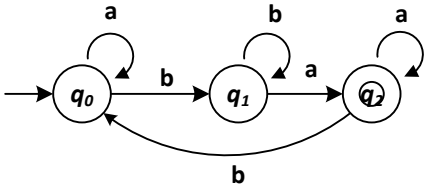
$[q_0, \mathbf{baba}] \vdash [q_1, aba]$
$\vdash [q_0, ba]$

$\vdash$ $[q_1 , a]$
$\vdash$ $[q_0 , \lambda]$
Stop at q_0 (final state) and all alphabet is traced (string is empty);
Hence, string **baba is accepted** by the machine.

c) *abbb*, *baab*, and *baba* are accepted. *bbb*, *baa* are not accepted.
d) ((a+b) (a+b))*

4.

a) Yes, DFA.



b)
$[q_0 , \mathbf{abaa}]$ $\vdash$ $[q_0 , baa]$
$\vdash$ $[q_1 , aa]$
$\vdash$ $[q_2 , a]$
$\vdash$ $[q_2 , \lambda]$
Stop at q_2 (final state) and all alphabet is traced (string is empty);
Hence, string **abaa is accepted** by the machine.

$[q_0 , \mathbf{bbbabb}]$ $\vdash$ $[q_1 , bbabb]$
$\vdash$ $[q_1 , babb]$
$\vdash$ $[q_1 , abb]$
$\vdash$ $[q_2 , bb]$
$\vdash$ $[q_0 , b]$
$\vdash$ $[q_1 , \lambda]$
All alphabet is traced (string is empty) but stop at q_1 (not final state) ;
Hence, string **bbbabb is not accepted** by the machine.

$[q_0 , \mathbf{bababa}]$ $\vdash$ $[q_1 , ababa]$
$\vdash$ $[q_2 , baba]$
$\vdash$ $[q_0 , aba]$
$\vdash$ $[q_0 , ba]$
$\vdash$ $[q_1 , a]$
$\vdash$ $[q_2 , \lambda]$
Stop at q_2 (final state) and all alphabet is traced (string is empty);
Hence, string **bababa is accepted** by the machine.

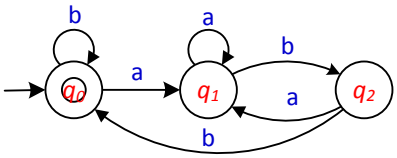
$[q_0 , \mathbf{bbbaa}]$ $\vdash$ $[q_1 , bbaa]$
$\vdash$ $[q_1 , baa]$
$\vdash$ $[q_1 , aa]$
$\vdash$ $[q_2 , a]$
$\vdash$ $[q_2 , \lambda]$
Stop at q_2 (final state) and all alphabet is traced (string is empty);
Hence, string **bbbaa is accepted** by the machine.

c) *abaa, bababa,* and *bbbaa* are accepted and *bbbabb* is not accepted.

d) a*bb*aa*(ba*bb*aa*)* and (a*bb*aa*b)*a*bb*aa* or $a^*b^+a^+(ba^*b^+a^+)^*$ and $(a^*b^+a^+b)^*a^*b^+a^+$, since $bb^* = b^+$

5.

a)



DFA

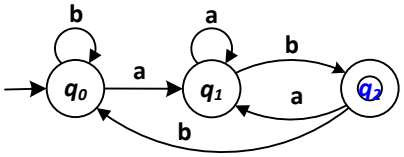
b) $[q_0, \textbf{babaab}] \vdash [q_0, abaab]$
$\vdash [q_1, baab]$
$\vdash [q_2, aab]$
$\vdash [q_1, ab]$
$\vdash [q_1, b]$
$\vdash [q_2, \lambda]$

All alphabet is traced (string is empty) but stop at q_2 (not final state) ;
Hence, string **babaab is not accepted** by the machine.

c) **(λ + b + (aa*(ba)*bb (b*aa*(ba)* bb)*)*)***

6.

a)



DFA

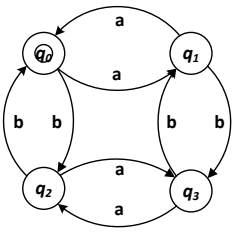
b) $[q_0, \textbf{babaab}] \vdash [q_0, abaab]$
$\vdash [q_1, baab]$
$\vdash [q_2, aab]$
$\vdash [q_1, ab]$
$\vdash [q_1, b]$
$\vdash [q_2, \lambda]$

Stop at q_2 (final state) and all alphabet is traced (string is empty);
Hence, string **babaab is accepted** by the machine.

c) b*aa*b ((aa*b) + (bb*aa*b))*

7.

a) $q_0 \rightarrow aq_1 \mid bq_2$
$q_1 \rightarrow aq_0 \mid bq_3$
$q_2 \rightarrow aq_3 \mid bq_0$
$q_3 \rightarrow aq_2 \mid bq_1$



DFA

The language consisting of all strings over {*a, b*} with an even number of *a*'s and an even number of *b*'s.

b) $[q_0 , \mathbf{ab}] \Rightarrow [q_1 , b]$
$\Rightarrow [q_3 , \lambda]$
All alphabet is traced (string is empty) but stop at q_3 (not final state) ;
Hence, string **ab is not accepted** by the machine.

$[q_0 , \mathbf{bb}] \Rightarrow [q_2 , b]$
$\Rightarrow [q_0 , \lambda]$
Stop at q_0 (final state) and all alphabet is traced (string is empty);
Hence, string **bb is accepted** by the machine.

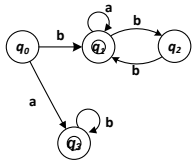
$[q_0 , \mathbf{aab}] \Rightarrow [q_1 , ab]$
$\Rightarrow [q_0 , b]$
$\Rightarrow [q_2 , \lambda]$
All alphabet is traced (string is empty) but stop at q_2 (not final state) ;
Hence, string **aab is not accepted** by the machine.

$[q_0 , \mathbf{baba}] \Rightarrow [q_2, aba]$
$\Rightarrow [q_3, ba]$
$\Rightarrow [q_1, a]$
$\Rightarrow [q_0 , \lambda]$
Stop at q_0 (final state) and all alphabet is traced (string is empty);
Hence, string **baba is accepted** by the machine.

c) Only *bb* and *baba*.
d) (a(bb)*a + b(aa)*b)*

8.

a)



No, NFA.

b) $[q_0 , \mathbf{abbb}] \vdash [q_3 , bbb]$
$\vdash [q_3 , bb]$
$\vdash [q_3 , b]$
$\vdash [q_3 , \lambda]$
Stop at q_3 (final state) and all alphabet is traced (string is empty);
Hence, string **abbb is accepted** by the machine.

$[q_0$, **bbb**$]$ $\vdash$ $[q_1$, bb$]$
$\vdash$ $[q_2$, b$]$
$\vdash$ $[q_1$, λ$]$
Stop at q_1 (final state) and all alphabet is traced (string is empty);
Hence, string **bbb is accepted** by the machine.

$[q_0$, **baa**$]$ $\vdash$ $[q_1$, aa$]$
$\vdash$ $[q_1$, a$]$
$\vdash$ $[q_1$, λ$]$
Stop at q_1 (final state) and all alphabet is traced (string is empty);
Hence, string **baa is accepted** by the machine.

$[q_0$, **baab**$]$ $\vdash$ $[q_1$, aab$]$
$\vdash$ $[q_1$, ab$]$
$\vdash$ $[q_1$, b$]$
$\vdash$ $[q_2$, λ$]$
All alphabet is traced (string is empty) but stop at q_2 (not final state) ;
Hence, string **baab is not accepted** by the machine.

$[q_0$, **bbbaa**$]$ $\vdash$ $[q_1$, bbaa$]$
$\vdash$ $[q_2$, baa$]$
$\vdash$ $[q_1$, aa$]$
$\vdash$ $[q_1$, a$]$
$\vdash$ $[q_1$, λ$]$
Stop at q_1 (final state) and all alphabet is traced (string is empty);
Hence, string **bbbaa is accepted** by the machine.

c) Only *abbb*, *bbb*, *baa* and *bbbaa* are accepted.
d) ab* + b (a + bb)*

9.

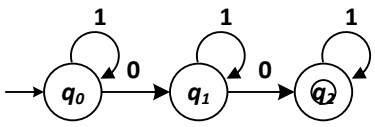
a)

δ	***a***	***b***
q_0	q_0	q_1
q_1	-	q_1

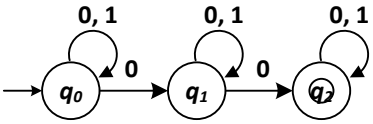
b) *abb, aab, aaaab* are accepted by M
c) *a*b**

10.

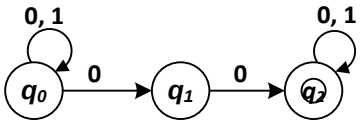
a) 1*01*01*



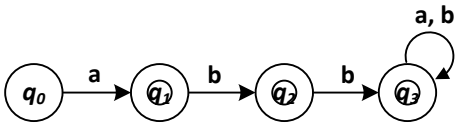
b) (0 + 1)*0(0 + 1)*0(0 + 1)*



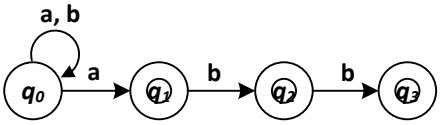
c) (0 + 1)*00(0 + 1)*



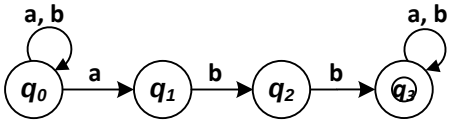
d) *abb*(*a* + *b*)*



e) (*a* + *b*)**abb*



f) (*a* + *b*)**abb*(*a* + *b*)*



11.

a)

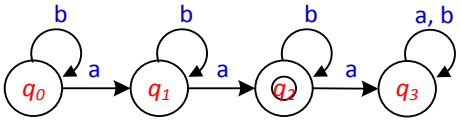
δ	*a*	*b*
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_0	q_1

b) Only *baab* and *abaaab* are accepted by M

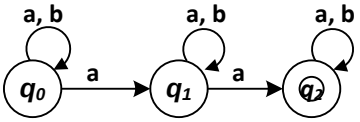
c) b*aa*b (ba*b + ab*aa*b)*

12.

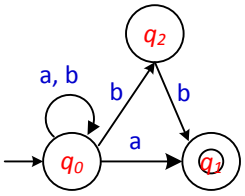
a) b*ab*ab*



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

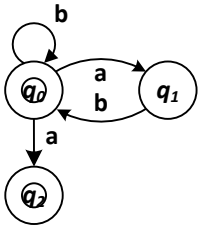


c) (a + b)*(a + bb)

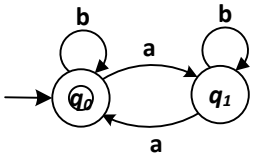


d) The language of all strings that begin or end with *aa* or *bb*. (aa + bb)(a + b)* + (a + b)* (aa + bb)

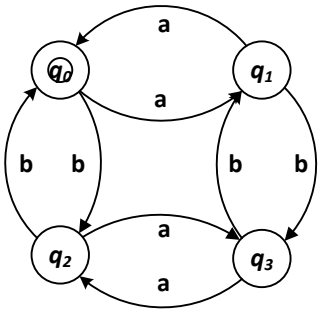
e) (b + ab)*(a + λ)



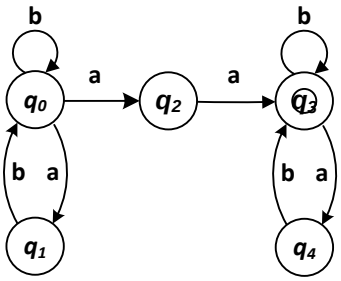
f) (ab*a + b)*



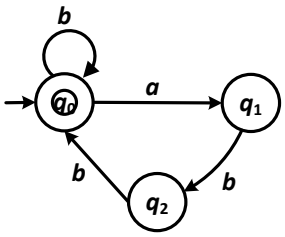
g)



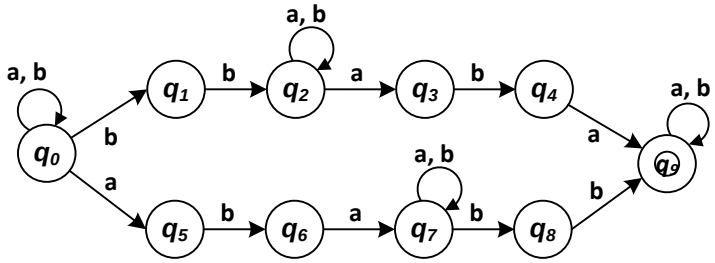
h) $(b + ab)^* aa(b + ba)^*$



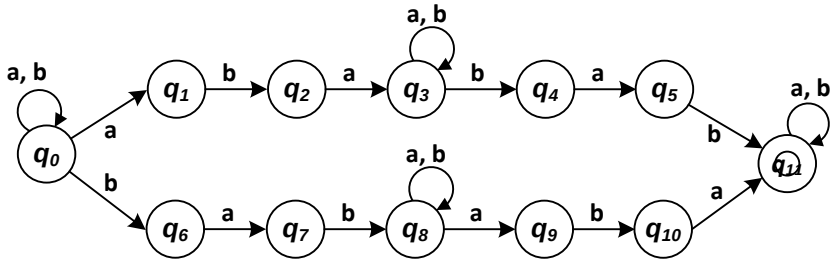
i) (abb + b)*



j) (a + b)*bb(a + b)*aba(a + b)* + (a + b)*aba(a + b)*bb(a + b)*



k) (a + b)*aba(a + b)*bab(a + b)* + (a + b)*bab(a + b)*aba(a + b)*



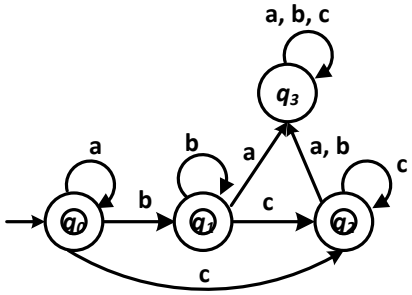
13. For each of the following languages, draw an FA accepting it.

a) $(a + b)^*a$
b) $(bb + ba)^*$
c) $(a + b)^*(b + aa)\ (a + b)^*$
d) $(bbb + baa)^*a$
e) $a + ba^* + ab^*a$
f) $(a + b)^*(ab + bba)$
g) $(b + bba)^*a$
h) $(aba + aa)^*(ba)^*$

14.

a) a*b*c*

The DFA



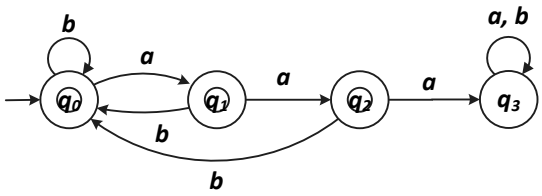
that accepts a*b*c* uses the loops in states q0, q1, and q2 to read a's, b's, and c's. Any deviation from the prescribed order causes the computation to enter q3 and reject the string.

b) The set of strings over {*a, b*} in which the substring *aa* occurs at least twice. **(a + b)*aa(a + b)*aa(a + b)***

c) The set of strings over {*a, b*} that do not begin with the substring *aaa*. (b + ab + aab)(a + b)*

d) (b ∪ ab ∪ aab)*(λ ∪ a ∪ aa)

A DFA that accepts the strings over {a, b} that do not contain the substring aaa is given by the state diagram



The states are used to count the number of consecutive a's that have been processed. When three consecutive a's are encountered, the DFA enters state q3, processes the remainder of the input, and rejects the string.

e) The set of strings over {*a, b, c*} that begin with *a,* contain exactly two *b*'s, and end with *cc.* *a(a + c)*b(a + c)*b(a + c)*cc*

f) The *set* of strings *over* {*a, b, c*} in which every *b* is immediately followed by at least one *c*. (a + bc + c)*

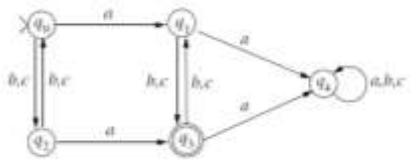
g) The set of strings over {*a, b*} in which the number of *a*'s is divisible by three.

h) The set of strings over {*a, b*} in which every *a* is either immediately preceded or immediately followed by *b,* for example, *baab, aba,* and *b.* **(aba + ba + ab + b)***

i) The set of strings of odd length over {*a, b*} that contain the substring *bb.*

j) The set of strings over {*a, b*} that have odd length or end with *aaa.*

k)



accepts strings of even length over {*a, b, c*} that contain exactly one a. A string accepted by this machine must have exactly one a and the total number of b's and c's must be odd. States q1 or q3 are entered upon processing a single a. The state q3 represents the combination of the two conditions required for acceptance. Upon reading a second a, the computation enters the nonaccepting state q4 and rejects the string.

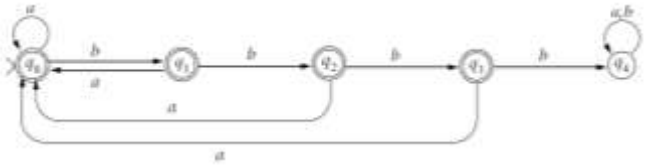
l) The set of strings over {*a, b*} that have an odd number of occurrences of the substring *aa.* Note that *aaa* has two occurrences of *aa.*

m) The set of strings over {*a, b*} that contain an even number of substrings *ba.*

n) The set of strings over {1, 2, 3} the sum of whose elements is divisible by six.

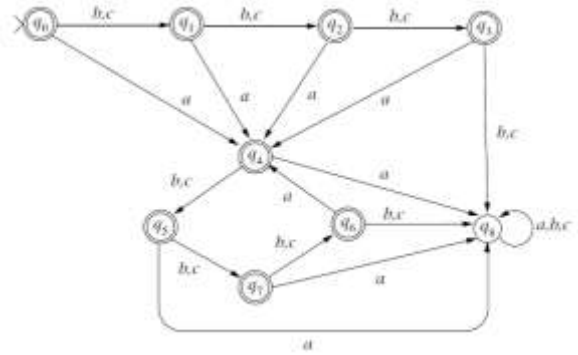
o) The set of strings over {*a, b, c*} in which the number of *a*'s plus the number of b's plus twice the number of *c*'s is divisible by six.

p) The states of the machine are used to count the number of symbols processed since an a has been read or since the beginning of the string, whichever is smaller.



A computation is in state qi if the previous i elements are not a's. If an input string has a substring of length four without an a, the computation enters q4 and rejects the string. All other strings accepted.

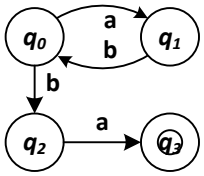
q) The DFA to accept this language differs from the machine in Exercise 20 by requiring every fourth symbol to be an a. The first a may occur at position 1, 2, 3, or 4. After reading the first a, the computation enters state q4 and checks for a pattern consisting of a combination of three b's or c's followed by an a. If the string varies from this pattern, the computation enters state q8 and rejects the input.



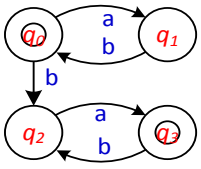
Strings of length 3 or less are accepted, since the condition is vacuously satisfied.

15.

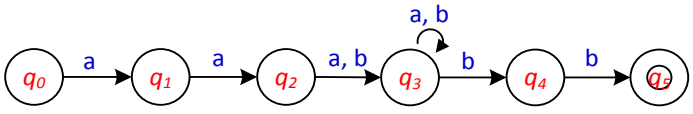
a)



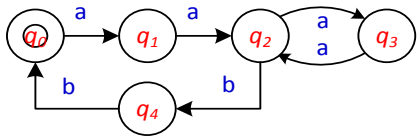
b)



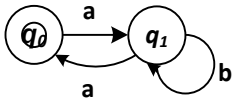
c)



d)



e)



16.

IN the language	NOT IN the language
c, ab, ac, aab, acc, acab, abab, abaab, cc, cab, caab, ccab, ccac, abacab	*a, b, bc, ca, cb, caa*

17.

a)

	a	b
q_0	$\{q_0, q_1\}$	Ø
q_1	Ø	$\{q_1, q_2\}$
q_2	$\{q_0, q_1\}$	Ø

b)

$[q_0 , aaabb] \Rightarrow [q_0 , aabb]$
$\Rightarrow [q_0 , abb]$
$\Rightarrow [q_0 , bb]$ No such input, string rejected

$[q_0 , aaabb] \Rightarrow [q_0 , aabb]$
$\Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_1 , b]$
$\Rightarrow [q_1 , \lambda]$ Not final state, string rejected

$[q_0 , aaabb] \Rightarrow [q_0 , aabb]$
$\Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_1 , b]$
$\Rightarrow [q_2 , \lambda]$ String Accepted

$[q_0 , aaabb] \Rightarrow [q_0 , aabb]$
$\Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_2 , b]$ No such input, string rejected

$[q_0 , aaabb] \Rightarrow [q_0 , aabb]$
$\Rightarrow [q_1 , abb]$ No such input, string rejected

$[q_0 , aaabb] \Rightarrow [q_1 , aabb]$ No such input, string rejected

c) Yes
d) a*ab*b(ab*b + aa*ab*b)* or $(a^+b^+)^+$.
A regular expressions for L(M) is $a^+b^+(ab^+ + a^+ab^+)^*$. Processing a^+b^+ leaves the computation in state q2. After arriving at q2 there are two ways to leave and return, taking the cycle q0, q1, q2 or the cycle q1, q2. The first path processes a string from a^+ab^+ and the second ab^+. This expression can be reduced to $(a^+b^+)^+$.

18.
a)

	a	b
q_0	$\{q_0, q_1\}$	Ø
q_1	Ø	$\{q_0, q_2\}$
q_2	Ø	$\{q_1, q_2\}$

b)

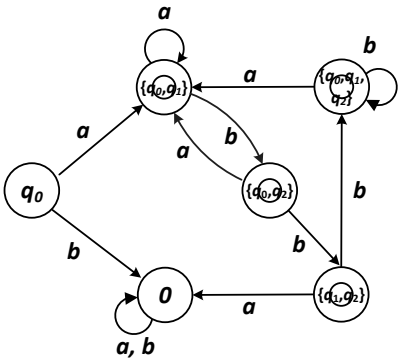
$[q_0 , aabb] \Rightarrow [q_0 , abb]$
$\Rightarrow [q_0 , bb]$ String Rejected

$[q_0 , aabb] \Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_2 , b]$
$\Rightarrow [q_1 , \lambda]$ String Accepted

$[q_0 , aabb] \Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_2 , b]$
$\Rightarrow [q_2 , \lambda]$ String Accepted

$[q_0 , aabb] \Rightarrow [q_0 , abb]$
$\Rightarrow [q_1 , bb]$
$\Rightarrow [q_0, b]$ String Rejected
…etc..!

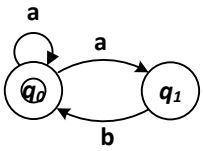
c) Yes
e) a*a(ba*a + bb*b)* stop at q_1
a*ab(b + bb + bba*ab)* stop at q_2
Regular exp is: a*a(ba*a + bb*b)* + a*ab(b + bb + bba*ab)* or
$a^+(ba^+ + bb^+)^* + a^+b(b + bb + bba^+b)^*$ since $a^*a = a^+$ and $b^*b = b^+$

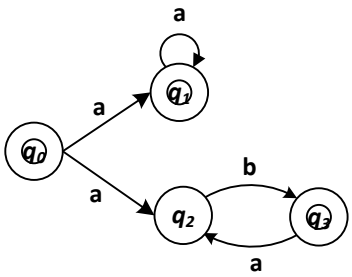


19.

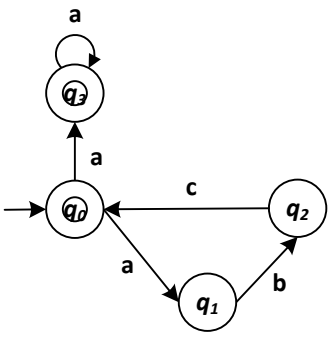
a) $(a \cup ab \cup aab)^*$

b)

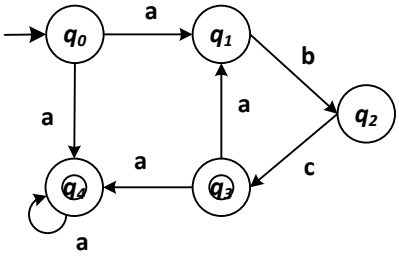




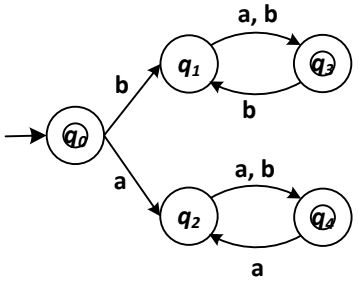
c) The NFA



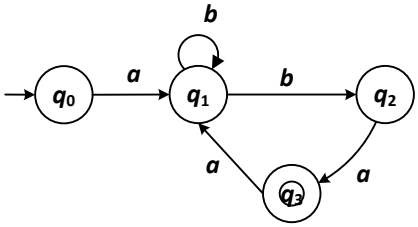
accepts the language (abc)*a*. Strings of the form (abc)* are accepted in q0. State q3 accepts $(abc)^*a^+$.



d)



e)



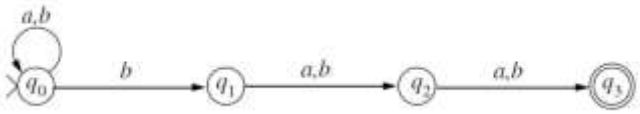
20.

a) The set of strings over {*a, b*} that contain either *aa* and *bb* as substrings. (a + b)*aa(a + b)*bb(a + b)* + (a + b)*bb(a + b)*aa(a + b)*

b) The set of strings over {*a, b*} that contain both or neither *aa* and *bb* as substrings.

c) The set of strings over {*a, b*} whose third-to-the-last symbol is *b.* (a + b)*b(a + b)(a + b)

The set of strings over fa; bg whose third to the last symbol is b is accepted by the NFA



When a b is read in state q0, nondeterminism is used to choose whether to enter state q1 or remain in state q0. If q1 is entered upon processing the third to the last symbol, the computation accepts the input. A computation that enters q1 in any other manner terminates unsuccessfully.

d) The set of strings over {*a, b*} whose third and third-to-last symbols are both *b.* For example, *aababaa, abbbbbbbb,* and *abba* are in the language. (a + b)(a + b)b(a + b)*b(a + b)(a + b) + (a + b)(a + b)b(a + b)(a + b) + (a + b)bb(a + b) + b(a + b)b

e) The set of strings over {*a, b*} in which every *a* is followed by *b* or *ab.* (ab + aab + b)

f) The set of strings over {*a, b*} that have a substring of length four that begins and ends with the same symbol. a(a + b)(a + b)a + b(a + b)(a + b)b

g) The set of strings over {*a, b*} that contain substrings *aaa* and *bbb.* (a + b)*aaa(a + b)*bbb(a + b)* + (a + b)*bbb(a + b)*aaa(a + b)*

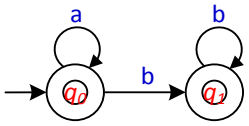
h) The set of strings over {*a, b, c*} that have a substring of length three containing each of the symbols exactly once. abc + acb + bac + bca + cba + cab

21. Construct the state diagram of a DFA that accepts the strings over {*a, b*} ending with the

substring *abba.* Give the state diagram of an NFA with six arcs that accepts the same language. (a + b)*abba

22.

a)



$S \rightarrow \quad aS \mid bA \mid \lambda$

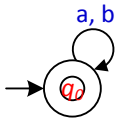
$A \rightarrow \quad bA \mid \lambda$

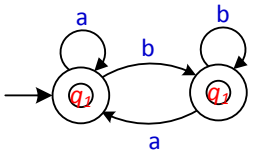
b) $q_0 \rightarrow \quad aq_0 \mid bq_0 \mid \lambda$

$S \rightarrow aA \mid bB \mid \lambda$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bB \mid aS \mid \lambda$





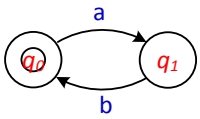
c) $q_0 \rightarrow \quad aq_1 \mid \lambda$

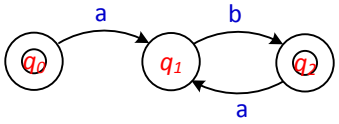
$q_1 \rightarrow \quad bq_0$

$S \rightarrow aA \mid \lambda$

$A \rightarrow bB$

$B \rightarrow aA \mid \lambda$



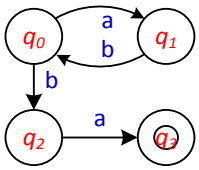


d) $q_0 \rightarrow \quad aq_1 \mid bq_2$

$q_1 \rightarrow \quad bq_0$

$q_2 \rightarrow \quad aq_3$

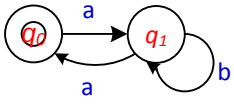
$q_3 \rightarrow \quad \lambda$



e)

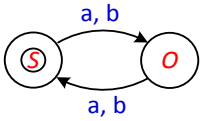
$q_0 \rightarrow \quad aq_1 \mid \lambda$

$q_1 \rightarrow \quad bq_1 \mid aq_0$



23.

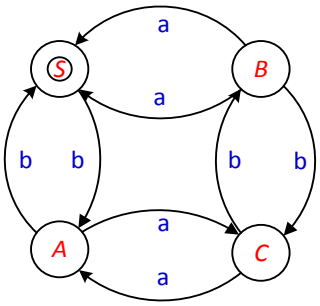
a)



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

24.

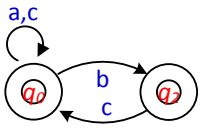
a)



b) aa, bb, abba, baab, baaaab, abab.

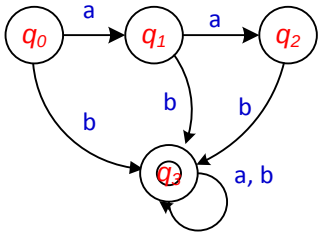
25.

a) RE = (a + bc + c)*



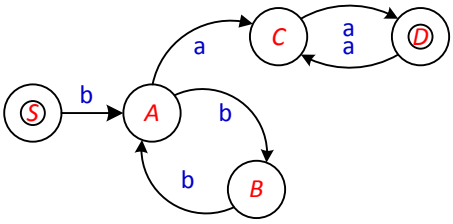
only q_0 final, remove final in q_1

b)



26.

a)



b) b(bb)*aa(aa)*

27. Design a finite automaton to accept the language defined by the regular expression

a) (ab)*(ba)*

b) $aa(a+b)^{+}bb$

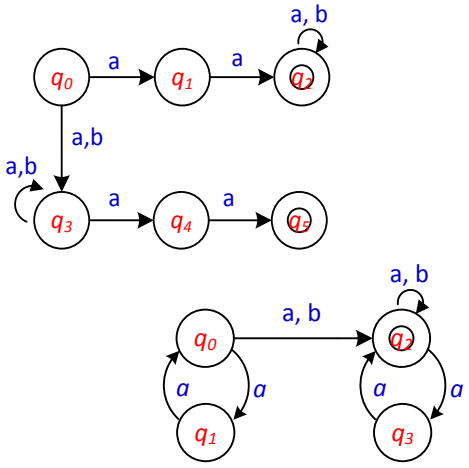
c) $((aa)^{+}bb)^{*}$

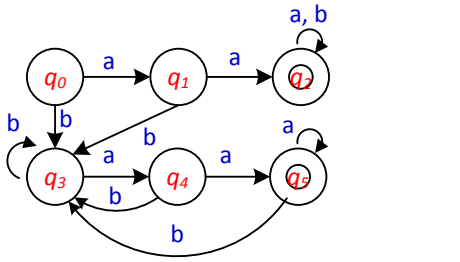
d) aa*b + aa(ba)*

e) b*a(ab*b)* + λ

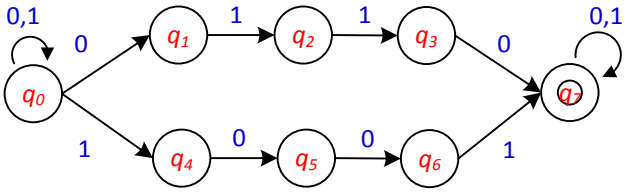
28.

a)





29.



30. ((a + b) (a + b)) (a + b)* ((a + b) (a + b))

31.

	DFA or NFA	No.
(0)	DFA	Z
(i)	DFA	j
(ii)	NFA	b
(iii)	NFA	a
(iv)	DFA	g
(v)	DFA	i
(vi)	DFA	h
(vii)	DFA	e
(viii)	DFA	f

32.

Table 1.	Table 2.
No.	Matching no. of RE
1.	**3**
2.	**8**
3.	**7**
4.	**6**
5.	**1**
6.	**2**
7.	**5**
8.	**9**
9.	**10**
10.	**4**

(ix)	DFA	c
(x)	DFA	d

Answer for Exercises of Chapter 5 (Questions at Page 190, Introduction to Theory of Computer Science: Definitions and Examples)

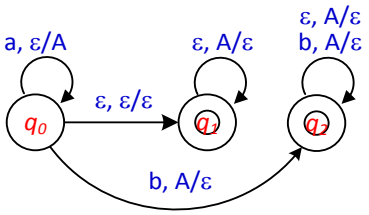
Part A Multiple Choices Questions

1. C
2. D
3. D
4. C
5. B

Part B Structured Questions

1.

a) The state diagram of M is



b) $(q_0, aab, \lambda) \vdash (q_0, ab, A) \vdash (q_0, b, AA) \vdash (q_2, \lambda, A) \vdash (q_2, \lambda, \lambda)$ accept
$(q_0, abb, \lambda) \vdash (q_0, bb, A) \vdash (q_2, b, \lambda)$ reject
$(q_0, aba, \lambda) \vdash (q_0, ba, A) \vdash (q_2, a, \lambda)$ reject

c) To show that the strings *aabb* and *aaab* are in L(M), we trace a computation of M that accepts these strings.

State	String	Stack
q_0	aabb	λ
q_0	abb	A
q_0	bb	AA
q_2	b	A
q_2	λ	λ

State	String	Stack
q_0	aaab	λ
q_0	aab	A
q_0	ab	AA
q_0	b	AAA
q_2	λ	AA
q_2	λ	A
q_2	λ	λ

Both of these computations terminate in the accepting state q2 with an empty stack.

d) The PDA M accepts the language $\{a^i b^j \mid 0 \le j \le i\}$. Processing an *a* pushes A onto the stack. Strings of the form a^i are accepted in state q_1. The transitions in q_1 empty the stack after the input has been read. A computation with input $a^i b^j$ enters state q_2 upon processing the first *b*. To read the entire input string, the stack must contain at least *j* A's. The transition $\delta(q_2, \lambda, A) = [q_2, \lambda]$ will pop any A's remaining on the stack.

The computation for an input string that has fewer a's than b's or in which an *a* occurs after a b halts in state q_2 without processing the entire string. Thus strings with either of these forms are rejected.

2.

a) $\delta(q_0, a, \lambda) = \{[q_0, A]\}$
$\delta(q_0, b, \lambda) = \{[q_0, B]\}$
$\delta(q_0, \lambda, \lambda) = \{[q_1, \lambda]\}$
$\delta(q_1, a, A) = \{[q_1, \lambda]\}$
$\delta(q_1, b, B) = \{[q_1, \lambda]\}$

b) $\{ww^R \mid w \in \{a, b\}^*\}$.

c) The computations of ab in M are as follows:

State	String	Stack
q0	ab	λ
q1	ab	λ

State	String	Stack
q0	ab	λ
q0	b	A
q1	b	A

State	String	Stack
q0	ab	λ
q0	b	A
q0	λ	BA
q1	λ	BA

The computations of abb in M are as follows:

State	String	Stack
q0	abb	λ
q1	abb	λ

State	String	Stack
q0	abb	λ
q0	bb	A
q1	bb	A

State	String	Stack
q0	abb	λ
q0	bb	A
q0	b	BA
q1	b	BA
q1	λ	A

State	String	Stack

q0	abb	λ
q0	bb	A
q0	b	BA
q0	λ	BBA
q1	λ	BBA

The computations of abbb in M are as follows:

State	String	Stack
q0	abbb	λ
q1	abbb	λ

State	String	Stack
q0	abbb	λ
q0	bbb	A
q1	bbb	A

State	String	Stack
q0	abbb	λ
q0	bbb	A
q0	bb	BA
q1	bb	BA
q1	b	A

State	String	Stack
q0	abbb	λ
q0	bbb	A
q0	bb	BA
q0	b	BBA
q1	b	BBA
q1	λ	BA

State	String	Stack
q0	abbb	λ
q0	bbb	A
q0	bb	BA
q0	b	BBA
q0	λ	BBBA
q1	λ	BBBA

d)

State	String	Stack
q0	aaaa	λ
q0	aaa	A
q0	aa	AA
q1	aa	AA
q1	a	A
q1	λ	λ

State	String	Stack
q0	baab	λ
q0	aab	B
q0	ab	AB
q1	ab	AB
q1	b	B

q1	λ	λ

e) To show that the string aaa and ab are not in L(M), we trace all computations of these strings in M, and check whether none of them accepts these strings. We have listed all the computations of ab in (b), and none of them accepts it. Now we trace all computations of aaa in M.

State	String	Stack
q0	aaa	λ
q1	aaa	λ

State	String	Stack
q0	aaa	λ
q0	aa	A
q1	aa	A
q1	a	λ

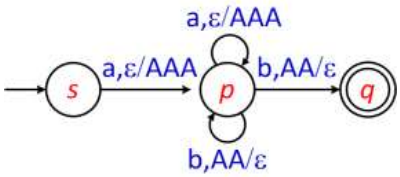
State	String	Stack
q0	aaa	λ
q0	aa	A
q0	a	AA
q1	a	AA
q1	λ	A

State	String	Stack
q0	aaa	λ
q0	aa	A
q0	a	AA
q0	λ	AAA
q1	λ	AAA

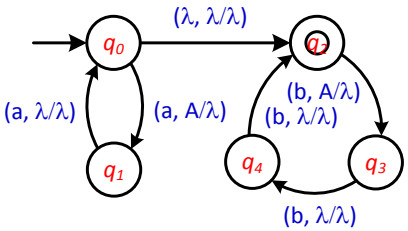
Since none of the computations above is accepted, we have aaa is not in M.

3.

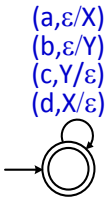
a)



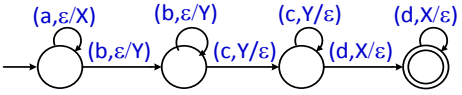
or



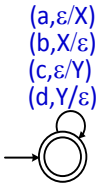
b)



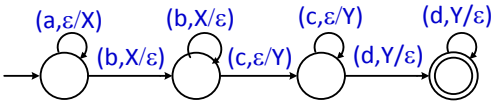
c)



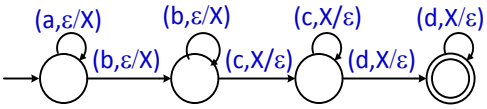
d)



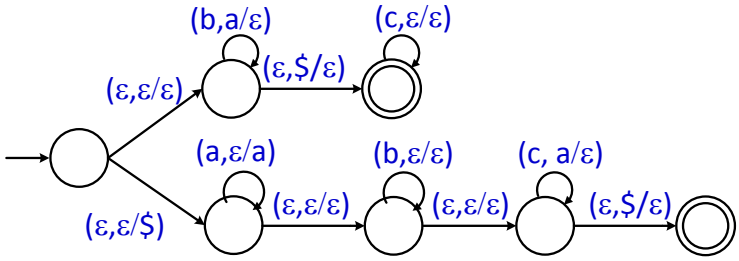
e)

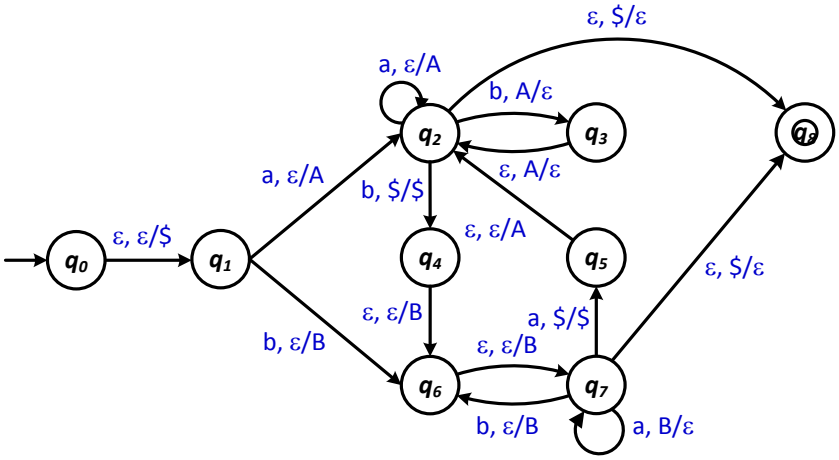


f)

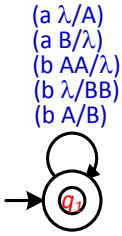


g)

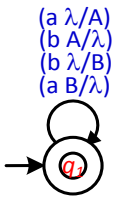




h)

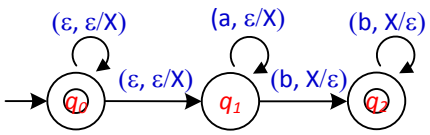


or

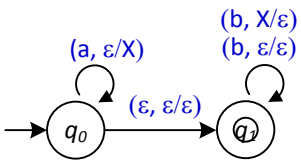


i)

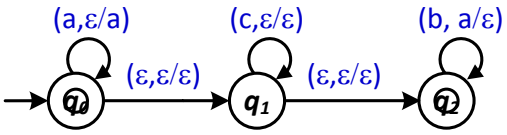
j)



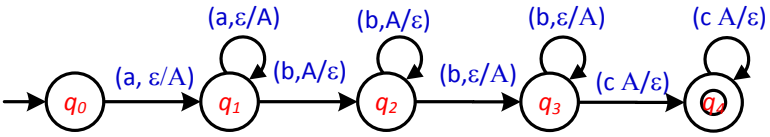
or



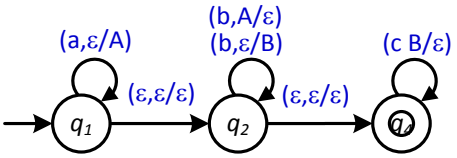
k)



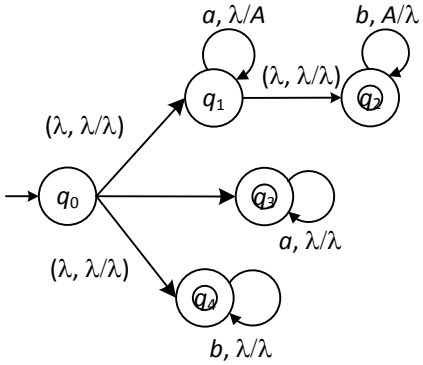
l)


(a,ε/A)
(b,A/ε)
(b,ε/A)
(c A/ε)
(a, ε/A)
(b,A/ε)
(b,ε/A)
(c A/ε)
q0
q1
q2
q3
q4

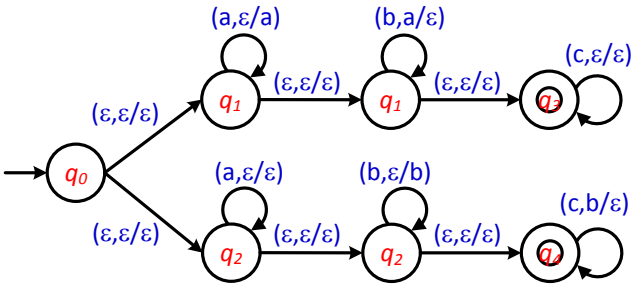
or


(a,ε/A)
(b,A/ε)
(b,ε/B)
(c B/ε)
(ε,ε/ε)
(ε,ε/ε)
q1
q2
q3

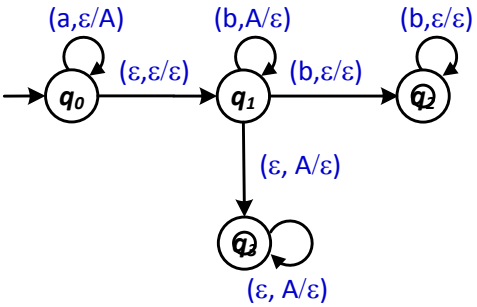
m)


a, λ/A
b, A/λ
(λ, λ/λ)
(λ, λ/λ)
(λ, λ/λ)
a, λ/λ
b, λ/λ
q0
q1
q2
q3
q4

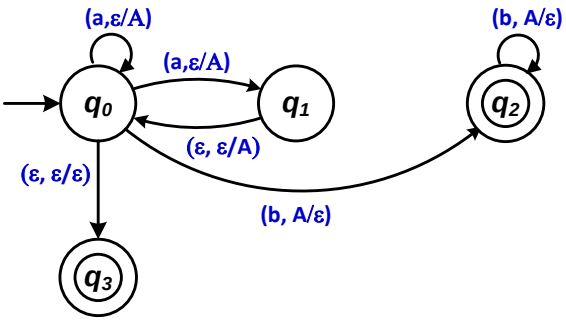
n)


(a,ε/a)
(b,a/ε)
(c,ε/ε)
(ε,ε/ε)
(ε,ε/ε)
(ε,ε/ε)
(a,ε/ε)
(b,ε/b)
(c,b/ε)
(ε,ε/ε)
(ε,ε/ε)
(ε,ε/ε)
q0
q1
q2
q3
q4

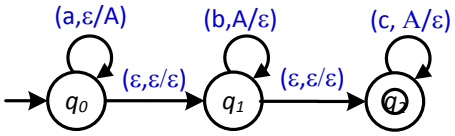
o)


(a,ε/A)
(b,A/ε)
(b,ε/ε)
(ε,ε/ε)
(b,ε/ε)
(ε, A/ε)
(ε, A/ε)
q0
q1
q2
q3

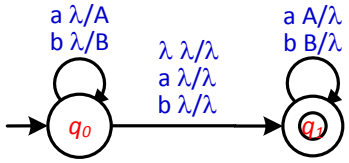
p)


(a,ε/A)
(b, A/ε)
(a,ε/A)
(ε, ε/A)
(ε, ε/ε)
(b, A/ε)
q0
q1
q2
q3

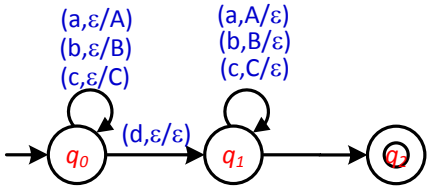
q)



r)

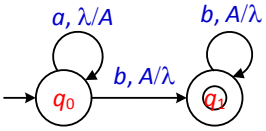


4.



5. Give the state diagram of a PDA M that accepts $\{a^{2i}b^{i+j} \mid 0 \le j \le i\}$ with acceptance by empty stack. Explain the role of the stack symbols in the computation of M. Trace the computations of M with input *aabb* and *aaaabb.*

6. The machine M



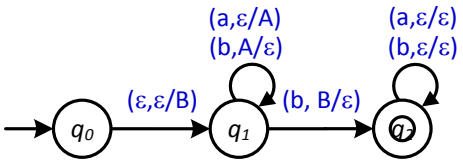
accepts the language L = $\{a^i b^j \mid i \ge 0\}$ by final state and empty stack.

a) Give the state diagram of a PDA that accepts L by empty stack.

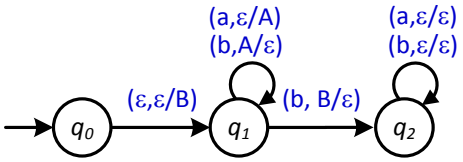
b) Give the state diagram of a PDA that accepts L by final state.

7.

a)



b)



8. Let L = $\{a^{2i}b^i \mid i \ge 0\}$.

a) Construct a PDA M_1 with $L(M_1) = L$.

b) Construct an atomic PDA M_1 with $L(M_2) = L$.

c) Construct an extended PDA M_3 with $L(M_3) = L$ that has fewer transitions than M_1.

d) Trace the computation that accepts the string *aab* in each of the automata constructed in (a), (b), and (c).

9. Let $L = \{a^{2i}b^{3i} \mid i \geq 0\}$.

a) Construct a PDA M_1 with $L(M_1) = L$.

b) Construct an atomic PDA M_2 with $L(M_2) = L$.

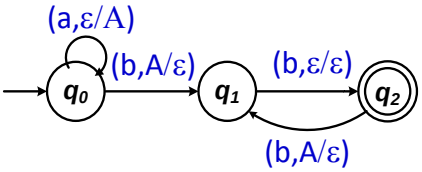
c) Construct an extended PDA M_3 with $L(M_3) = L$ that has fewer transitions than M_1.

d) Trace the computation that accepts the string *aabbb* in each of the automata constructed in (a), (b), and (c).

10.

a) This language accepts the strings which start with *a* and followed by *b*. The number of *b*'s is double than *a*'s.

b)



c) $L = \{ a^n b^{2n} \mid n > 0\}$

d) $S \rightarrow aSbb \mid aAbb$

$A \rightarrow \lambda$

e) $(q_0, aabbbb, \lambda) \vdash (q_0, abbbb, A) \vdash (q_0, bbbb, AA) \vdash (q_1, bbb, A)$
$\vdash (q_2, bb, A) \vdash (q_1, b, \lambda) \vdash (q_2, \lambda, \lambda)$ accept

f) $S \Rightarrow aSbb \Rightarrow aaSbbbb \Rightarrow aaAbbbb \Rightarrow aa\lambda bbbb$

11.

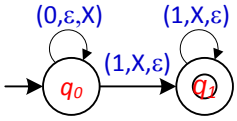
a) $(q_0, bbcbb, \lambda) \vdash (q_0, bcbb, A) \vdash (q_0, cbb, AA) \vdash (q_1, bb, AA)$
$\vdash (q_1, b, A) \vdash (q_1, \lambda, \lambda)$ accept

b) $S \rightarrow aS \mid bS \mid cA$

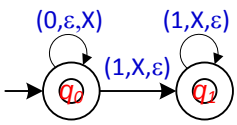
$A \rightarrow aA \mid bA \mid \lambda$

c) $S \Rightarrow bS \Rightarrow bbS \Rightarrow bbcA \Rightarrow bbcbA \Rightarrow bbcbbA \Rightarrow bbcb\lambda \Rightarrow bbcbb$

12. a)



b)

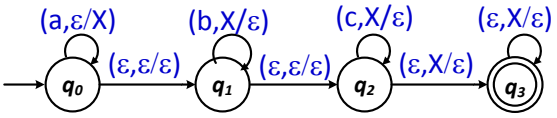


13. a) $(q_0, aaabbb, \lambda) \vdash (q_0, aabbb, A) \vdash (q_0, abbb, AA) \vdash (q_0, bbb, AAA)$
$\vdash (q_1, bb, AA) \vdash (q_1, b, A) \vdash (q_1, \lambda, \lambda)$ accept

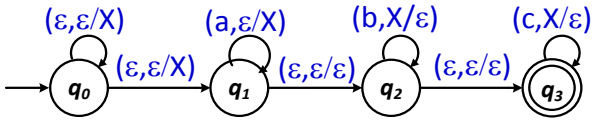
b) $(q_0, abb, \lambda) \vdash (q_0, bb, A) \vdash (q_1, bb, A) \vdash (q_1, b, \lambda)$ reject

14.

a)



b)



15. a)

CFG : $S \rightarrow XC \mid AY$

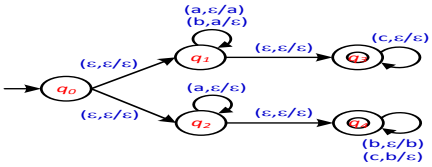
$X \rightarrow aXb \mid \varepsilon$

$C \rightarrow cC \mid \varepsilon$

$A \rightarrow aA \mid \varepsilon$

$Y \rightarrow bYc \mid \varepsilon$

b)

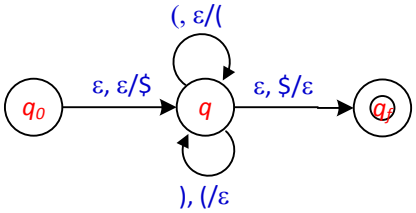


16.

- On seeing a (push it on the stack
- On seeing a) pop a (from the stack
- If attempt to pop an empty stack, reject
- If stack not empty at the end, reject
- Else accept

OR

- First push a “”bottom-of-the-stack” sysmbol $ and move to q_0
- On seeing a (push it onto the stack
- On seeing a) pop if a (is in the stack
- Pop $ and move to final state q_f



17.

a)

$S \rightarrow aABA \mid aBB$	$\delta(q0, a, \lambda) = \{[q0, ABA]^1, [q1, BB]^2\}$
$A \rightarrow bA \mid b$	$\delta(q1, b, A) = \{[q1, A]^3, [q1, \lambda]^4\}$
$B \rightarrow cB \mid c$	$\delta(q1, c, B) = \{[q1, B]^5, [q1, \lambda]^6\}$

b) Draw the PDA machine *M* that accept L(G).

M: $Q = \{q_0, q_1\}$

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

$\Gamma = \{A, B\}$

$F = \{q_1\}$

$\delta(q0, a, \lambda) = \{[q0, ABA]\}$

$\delta(q0, a, \lambda) = \{[q1, BB]\}$

$\delta(q1, b, A) = \{[q1, A]\}$

$\delta(q1, b, A) = \{[q1, \lambda]\}$

$\delta(q1, c, B) = \{[q1, B]\}$

$\delta(q1, c, B) = \{[q1, \lambda]\}$

18.

(i)	c
(ii)	i
(iii)	k
(iv)	j
(v)	e
(vi)	g
(vii)	a
(viii)	h
(ix)	f
(x)	b
(xi)	d

Answer for Exercises (Page 216, Theory of Computer Science: Definitions and Examples. 3rd Ed.)

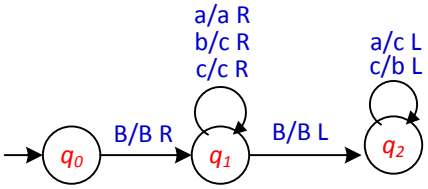
Part A Multiple Choices Questions

1. D
2. A
3. D

Part B Structured Questions

1.

a)



b) q_0BaabcaB
- ⊢ B$q_1$aabcaB
- ⊢ Ba$q_1$abcaB
- ⊢ Baa$q_1$bcaB
- ⊢ Baac$q_1$caB
- ⊢ Baaccq_1aB
- ⊢ Baaccaq_1B
- ⊢ Baaccq_2aB
- ⊢ Baac$q_2$ccB
- ⊢ Baa$q_2$cbcB
- ⊢ Ba$q_2$abbcB
- ⊢ B$q_2$acbbcB
- ⊢ q_2BccbbcB

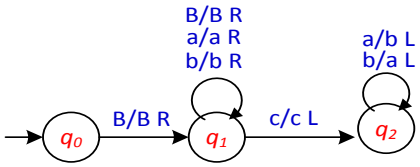
c) q_0BbcbcB
- ⊢ B$q_1$bcbcB
- ⊢ Bc$q_1$cbcB
- ⊢ Bcc$q_1$bcB
- ⊢ Bcccq_1cB
- ⊢ Bccccq_1B
- ⊢ Bcccq_2cB
- ⊢ Bcc$q_2$cbB
- ⊢ Bc$q_2$cbbB

⊢ Bq_2cbbbB
⊢ $q_2Bbbbbb$

d) The result of a computation is to replace the a's in the input string with c's and the c's with b's.

2.

a)



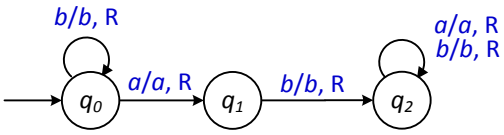
b) $q_0BabcabB$
⊢ $Bq_1abcabB$
⊢ Baq_1bcabB
⊢ $Babq_1cabB$
⊢ Baq_2bcabB
⊢ $Bq_2aacabB$
⊢ $q_2BbacabB$
halt

c) $q_0BbcbcB$
⊢ Bq_1bcbcB
⊢ Bbq_1cbcB
⊢ Bq_2bcbcB
⊢ $q_2BacbcB$
halt

d) The result of a computation is to replace the a's that appear before the first *c* in the input string with b's and the b's with a's.

3.

a)



b)

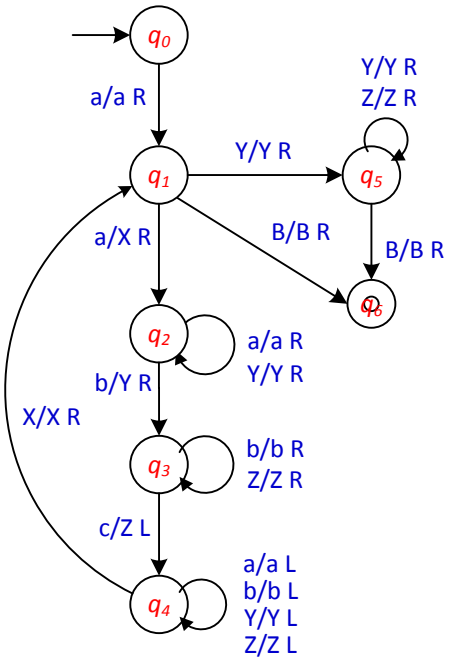
δ	a	b
q_0	q_1, a, R	q_0, b, R
q_1	-	q_2, b, R
q_2	q_2, a, R	q_2, b, R

c)

c) q_0abab
├ aq_1bab
├ abq_2ab
├ $abaq_2b$
├ $ababq_2$
halt

4.

a)



b)

Bq_0aabcB
├ Baq_1abcB
├ $BaXq_2bcB$
├ $BaXYq_3cB$
├ $BaXq_4YZB$
├ Baq_4XYZB
├ $BaXq_1YZB$
├ $BaXYq_5ZB$
├ $BaXYZq_5B$
├ $BaXYZBq_6$
Halt and accept.

$Bq_0aabbcB$
├ Baq_1abbcB
├ $BaXq_2bbcB$

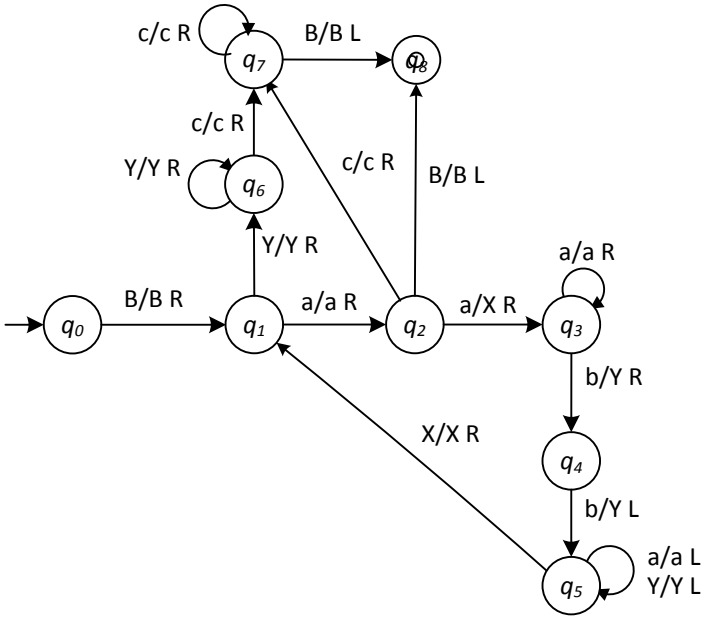
├ $BaXYq_3bcB$
├ $BaXYbq_3cB$
├ $BaXYq_4bZB$
├ $BaXq_4YbZB$
├ Baq_4XYbZB
├ $BaXq_1YbZB$
├ $BaXYq_5bZB$

Halt and reject.

c) a

5.

a)



b) Trace the computation of the Turing machine that process the string *aabbcc***.**

$q_0BaabbccB$

├ $Bq_1aabbccB$
├ $Baq_2abbccB$
├ $BaXq_3bbccB$
├ $BaXYq_4bccB$
├ $BaXq_5YYccB$
├ $Baq_5XYYccB$
├ $BaXq_1YYccB$
├ $BaXYq_6YccB$
├ $BaXYYq_6ccB$
├ $BaXYYcq_7cB$
├ $BaXYYccq_7B$
├ $BaXYYcq_8cB$

Halt and accept.

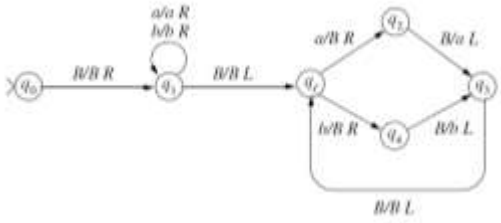
c) a

6.

a)

Answer from Sudkamp

Starting with the rightmost symbol in the input and working in a right-to-left manner, the machine diagrammed below moves each symbol one position to the right.

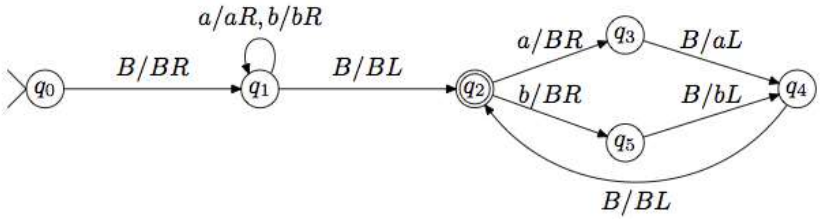


On reading the first blank following the input, the head moves to the left. If the input is the null string, the computation halts in state qf with the tape head in position zero as desired. Otherwise, an a is moved to the right by transitions to states q2 and q3.

Similarly, states q4 and q3 shift a *b*. This process is repeated until the entire string has been shifted.

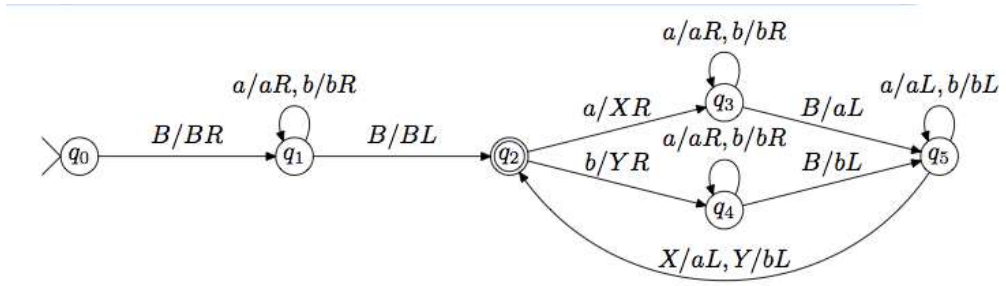
Answer from others:

First the head moves to the first blank following the input by transitions to states q0 and q1, and then moves to the left. At the state q2, starting with the rightmost symbol in the input and working in a right-to-left manner, the machine moves each symbol one position to the right.



If the input is the null string, the computation halts in the final state q2 with the tape head in position zero as desired. Otherwise, an a is moved to the right by transitions to states q3 and q4. Similarly, states q5 and q4 shift a b. This process is repeated until the entire string has been shifted.

b) First the head moves to the rightmost symbol in the input. At state q2, the machine works in a right-to-left manner, and copy a symbol to the end of the current string.



c) A Turing machine to insert blanks between the symbols in the input string can iteratively use the machine from Exercise 3 a) to accomplish the task. The strategy employed by the machine is

i. Mark tape position 0 with a #.

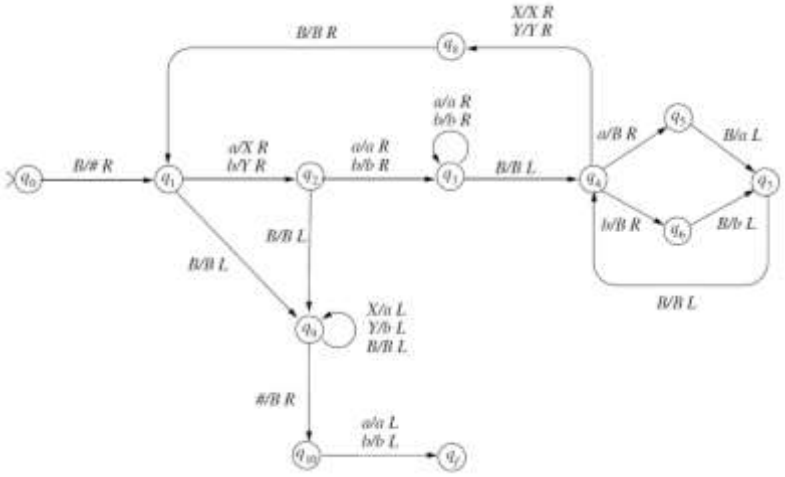
ii. If the input string is not empty, change the ¯rst symbol to X or Y to record if it was an a or b, respectively.

iii. Move to the end of the string and use strategy from the machine in part a) to move the unmarked input on square to the right.

iv. If there are no unmarked symbols on the tape, change all of the X's to a's and Y's to b's, and halt in position 0.

v. Otherwise, repeat the process on the unmarked string.

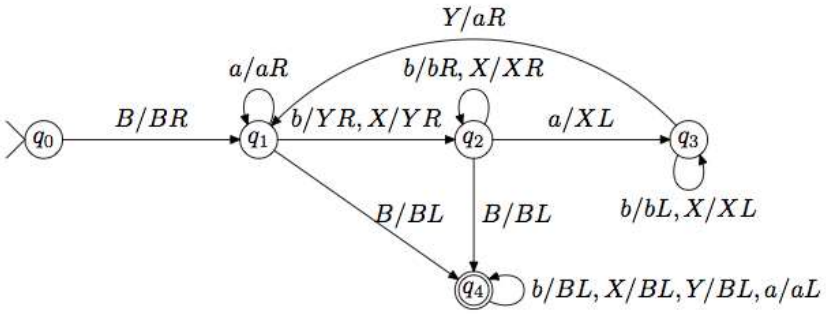
A Turing machine that accomplishes these tasks is



Processing the null string uses the arc from q1 to qf. Shifting a string one square to the right is accomplished by states q3 to q7. The arc from q4 to q8 when the shift has been completed. If another shift is needed, the entire process is repeated by entering q1.

d) The machine works in a left-to-right manner. At state q1, it skips a's and whenever it reads a b or a previously deleted position, denoted by symbol X, the machine move the first a in the following string. If no further a is find, it goes to the final state q4 and change all the remaining symbols b, X and Y

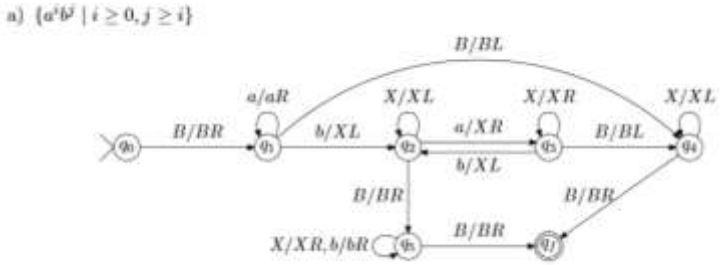
into blanks.



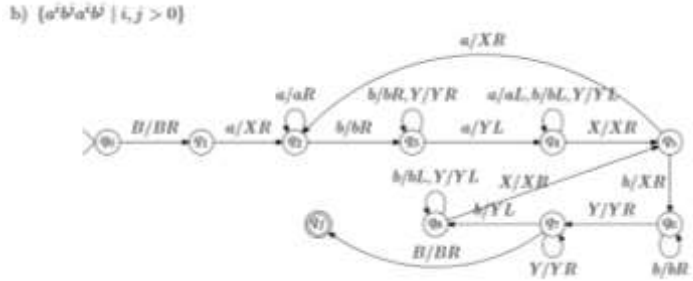
7. Construct a Turing machine with input alphabet {*a, b,* c} that accepts strings in which the first *c* is preceded by the substring *aaa.* A string must contain a *c* to be accepted by die machine.

8.

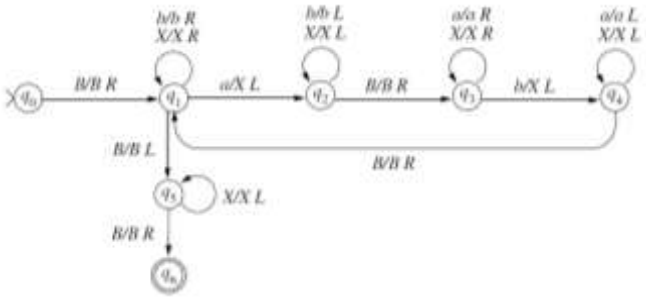
a)



b)



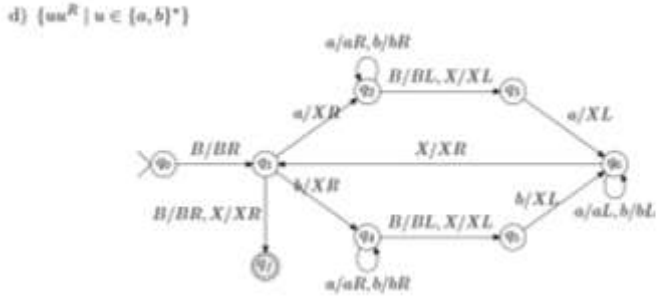
c) A computation of



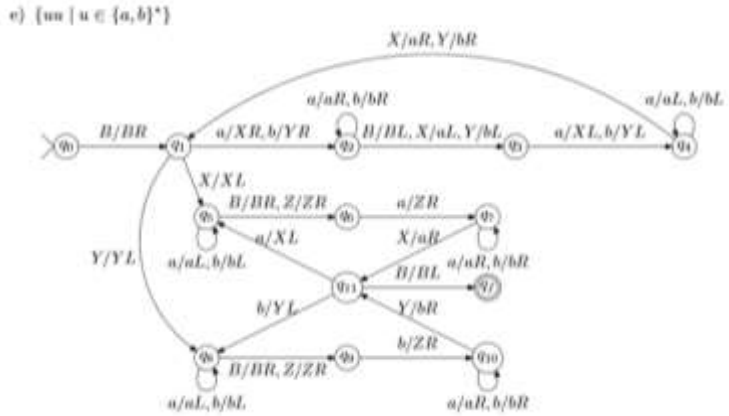
begins by finding the first a on the tape and replacing it with an X (state q1). The tape head is then returned to position zero and a search is initiated for a corresponding b. If a b is encountered in state q3, an X is written and the tape head

is repositioned to repeat the cycle q1, q2, q3, q4. If no matching b is found, the computation halts in state q3 rejecting the input. After all the a's have been processed, the entire string is read in q1 and q5 is entered upon reading the trailing blank. The computation halts in the accepting state q6 if no b's remain on the tape.

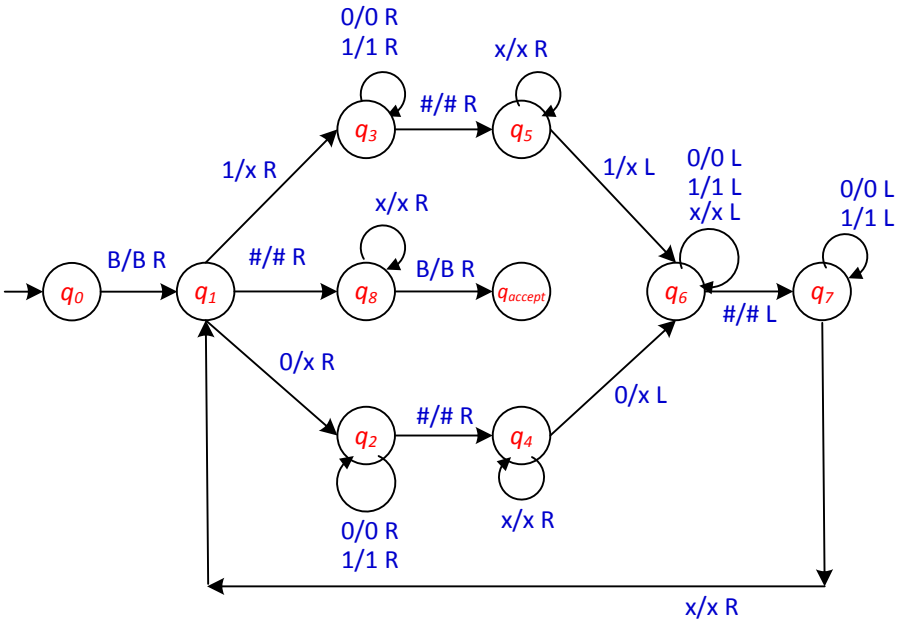
d)



e)



9.



10. Construct a Turing machine that performs some elementary arithmetic. It decides the language $C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

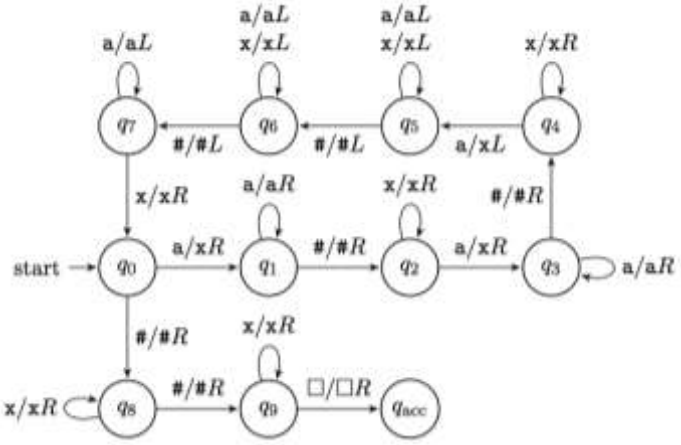
11. The Turing Machine (TM) for L_1 works in stages, where in every stage the first remaining *a* in every block is crossed off (i.e., replaced by an *x*). If at any stage in the process, there are no *a*'s left to cross off in some block, the TM rejects. If after all the *a*'s in the first block have been crossed off there are still some uncrossed *a*'s in other blocks, the TM rejects also.

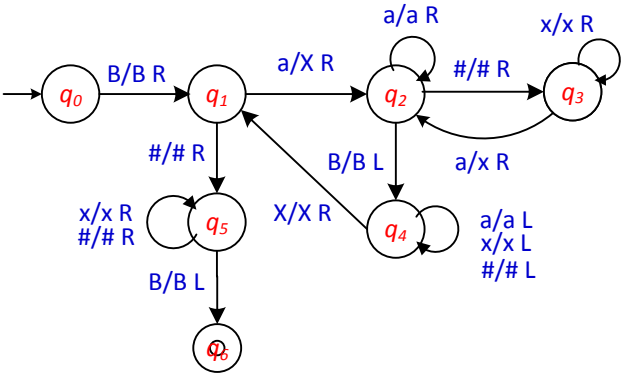
Here is a **high-level description** of the Turing Machine:

L_1 = "On input w :

a. Scan the input from left to right to cross off the leftmost a before the first # then cross off the first *a* we see after every #.
b. When there are no a's remaining in the first block: If there are no as left on the tape, accept, otherwise reject.
c. Return the head to the left-hand end of the tape and repeat stage 1. "

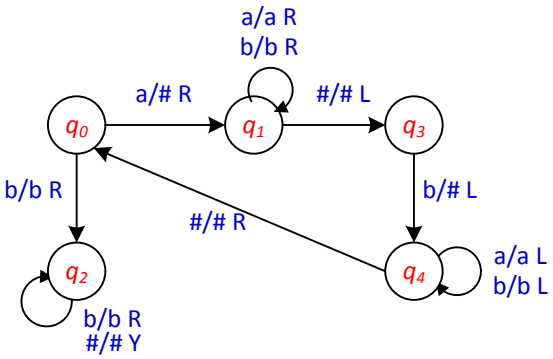
Here is a state diagram for L_1. The loop going around states q_0, q_1, q_2, and q_3 implements one stage of the crossing process. For the first block, a special cross off symbol â is used. This allows the Turing Machine to detect when it returns to the first block. After all the *a*'s in the first block have been crossed off, the TM goes into state q_4, which verifies that there are no *a*'s remaining anywhere on the tape.





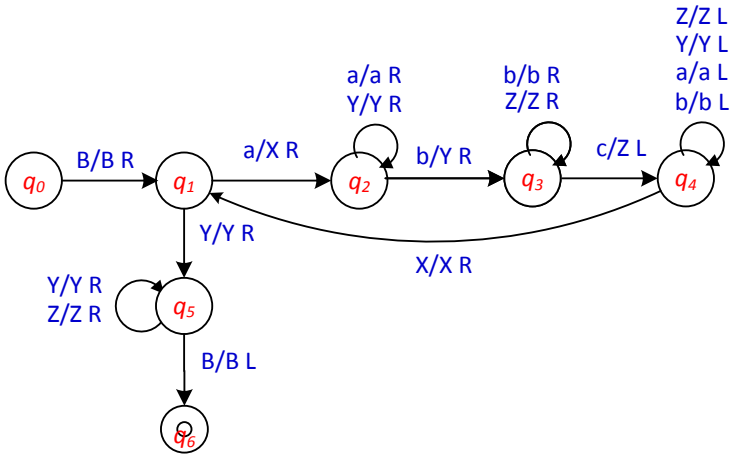
OR

12. Construct a Turing machine to accept the language $\{a^i b^j \mid i < j\}$

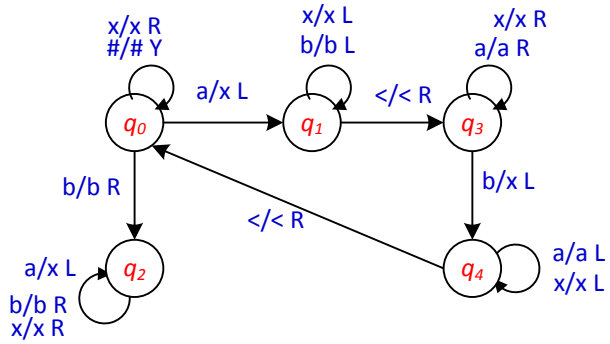


Y – Accept state

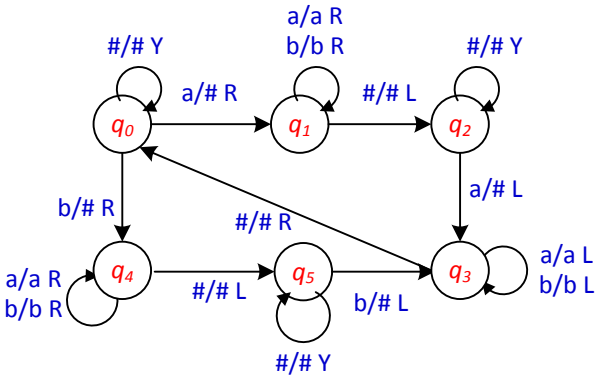
13.



14.



15.



16.

Example of Analysis For The Answer Of Question b

$M_2 = \langle Q, \Sigma, \Gamma, \delta, q_1, q_a, q_r \rangle$ is the TM that decides the language B = **{*w*#*w* | *w* ∈ {0, 1}*}**

$Q = \{ q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_a, q_r \}$

$\Sigma = \{0, 1, \#\}$

$\Gamma = \{0, 1, \#, x, \lozenge\}$

δ is described in Figure 2

The start, accept, reject are q_1, q_a, q_r, respectively

High-level description of M_2

***M*₂** = "On input w:

1. Scan the input tape to be sure that it contains a single #. If not, *reject.*
2. Zig-zag across the tape to corresponding positions on either side of # to check whether these positions contain the same symbol. If they do not, *reject*. Cross off the symbols as they are checked
3. When all symbols to the left of # have been crossed off, check for the remaining symbols to the right of #. If any symbol remain, *reject*; otherwise *accept*"

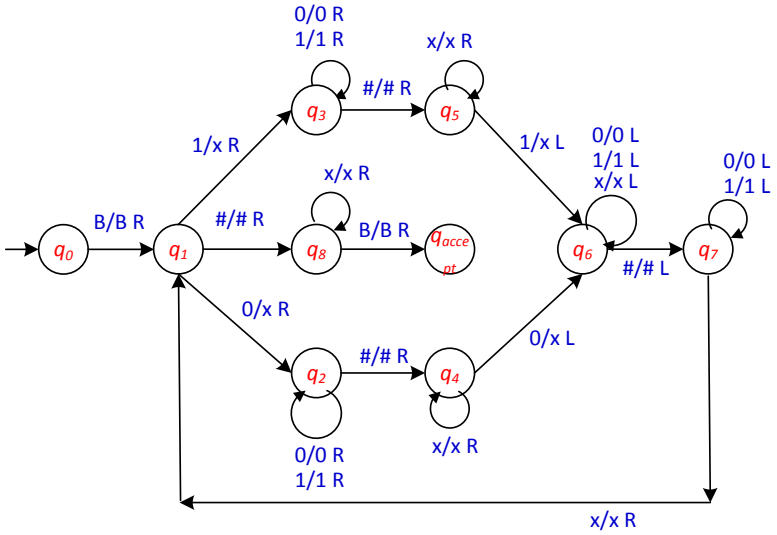
Note: High-level descriptions of TM-s are also called *implementation descriptions*.

[images]

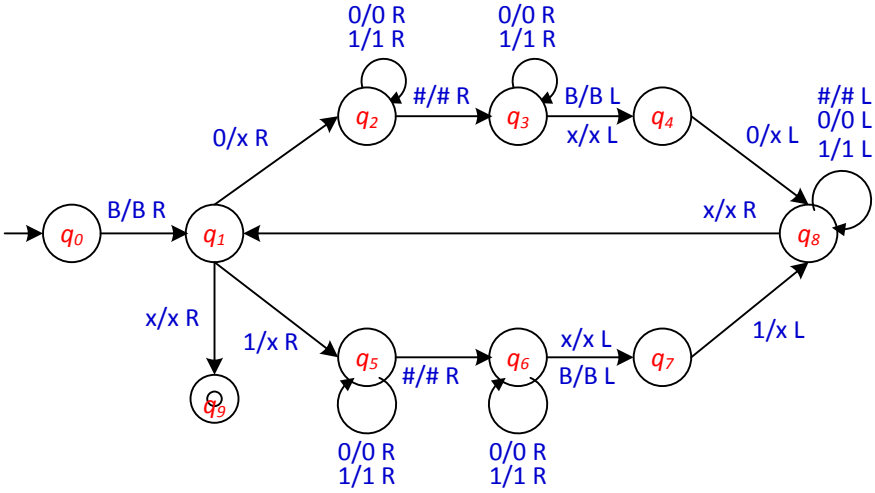
Transitions 0, 1 → *R* in states q_2 and q_3 means that machine moves to the right as long as 0 or 1 is on the tape. The machine starts by writing a blank symbol to delimit the left-hand edge of the tape

Stage 1 is implemented by states q_1 through q_7: q_2, q_4, q_6 if the first symbol of input is 0, and q_3, q_5, q_7 if the first input symbol was 1. To simplify the figure we don't show the reject state or transitions going to reject state. These transitions occur implicitly whenever a state lacks an outgoing transition for a particular symbol. Example, q_5 on # is such a transition.

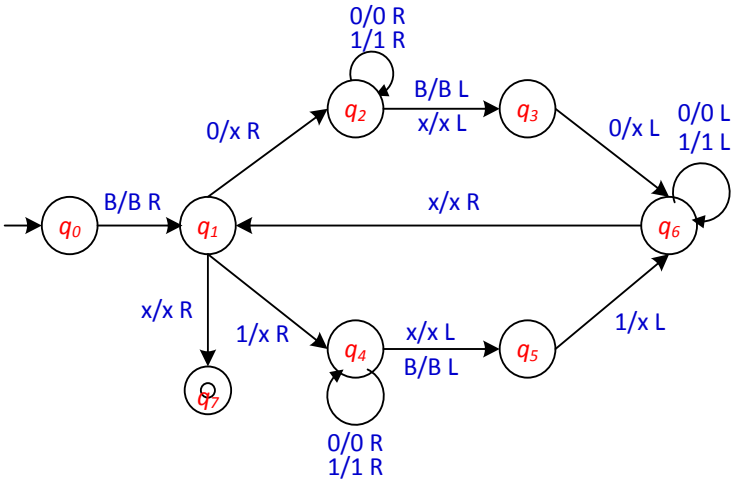
OR



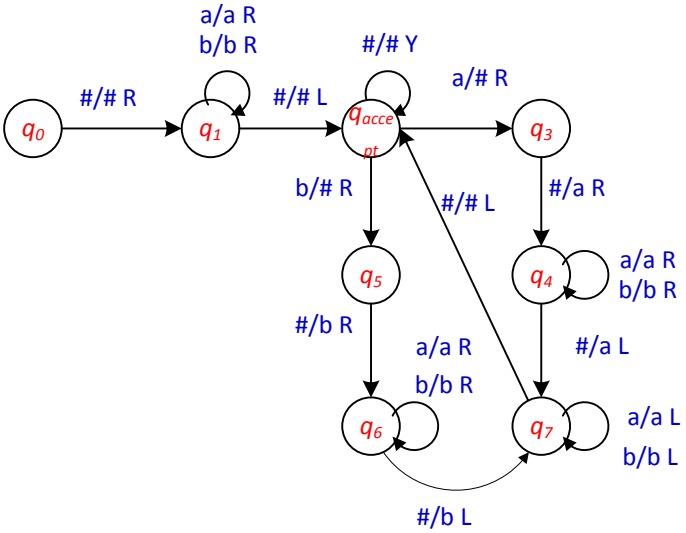
17.



18.



19.



21.

a) The final configuration is: (q2, ##abba#)

b) The final configuration is: (q2, ##baaaab#)

c) For an arbitrary input string w in {a,b}*, starting in the initial configuration (q0, #w#), this Turing machine halts with configuration (q2, ##$\mathbf{ww^R}$#). The machine takes an input string of the form *#w#* and returns an output string of the form *##ww^R#*

22.

a)

(q0,abaa#)	(q3,ABAA#)
(q1,Abaa#)	(q0,ABAA#)
(q1,Abaa#)	(q4,ABAA#)
(q1,Abaa#)	(q4,AbAA#)
(q1,Abaa#)	(q4,<abAA#)

(q2,Abaa#)	(q5,abAA#)
(q3,AbaA#)	(q7,AbAA#)
(q3,AbaA#)	(q7,AbAA#)
(q3,AbaA#)	(q8,Ab#A#)
(q0,AbaA#)	(q8,Ab#A#)
(q1,ABaA#)	(q5,Ab#A#)
(q1,ABaA#)	(q6,AB#A#)
(q2,ABaA#)	(q6,AB#A#)
	Crash

b)

(q0,abab#)	(q4,ABAB#)
(q1,Abab#)	(q4,AbAB#)
(q1,Abab#)	(q4,<abAB#)
(q1,Abab#)	(q5,abAB#)
(q1,Abab#)	(q7,AbAB#)
(q2,Abab#)	(q7,AbAB#)
(q3,AbaB#)	(q8,Ab#B#)
(q3,AbaB#)	(q8,Ab#B#)
(q3,AbaB#)	(q5,Ab#B#)
(q0,AbaB#)	(q6,AB#B#)
(q1,ABaB#)	(q6,AB#B#)
(q1,ABaB#)	(q8,AB###)
(q2,ABaB#)	(q8,AB###)
(q3,ABAB#)	(q5,AB###)
(q0,ABAB#	Accept

c) $\{ww \mid w \in \{a,b\}^*\}$

23.

a)

0	*<u>b</u>aaab#*
4	*#<u>a</u>aab#*
4	*#a<u>a</u>ab#*
4	*#aa<u>a</u>b#*
4	*#aaa<u>b</u>#*
4	*#aaab<u>#</u>*
5	*#aaa<u>b</u>#*
3	*#aa<u>a</u>##*
3	*#a<u>a</u>a##*

3 #a̲aa##
3 #a̲aa##
0 #a̲aa##
1 ##a̲a##
1 ##aa̲#
1 ##aa#̲#
2 ##a̲a##
3 ##a̲###
3 ##a̲###
0 ##a̲###
1 ######
2 ######̲ accepts

b) The Turing machine accepts the language $\{w \in \{a,b\}^* \mid w = w^R\}$

24.

a) Draw a state diagram for machine *M*.

b)

(0, #a̲b####)
|– (1, #a̲b####)
|– (1, #ab̲####)
|– (1, #ab#̲###)
|– (2, #ab̲####)
|– (5, #a#̲####)
|– (6, #a#b̲###)
|– (7, #a#b̲b##)
|– (7, #a#̲bb##)
|– (2, #a̲#bb##)
|– (3, ##a̲bb##)
|– (4, ##ab̲b##)
|– (4, ##abb̲##)
|– (4, ##abb#̲#)
|– (7, ##abb̲a#)
|– (7, ##abb̲a#)
|– (7, ##a̲bba#)
|– (7, ##abba#)
|– (2, ##a̲bba#) Y: accepts

c) The machine takes an input string of the form #̲*w*# and returns an output string of the form #̲#*ww*^{R}#

Discuss the differences between Finite Automata, Pushdown Automata and Turing Machine in terms of the following issues. Give example(s) if necessary.

	Finite Automata	**Pushdown Automata**	**Turing Machine**
Transition label in the state diagram			
Transition functions			
Configurations			
Computations (Way to describe the computation of the automata, for example, What state the FA is currently in, and what string is left to process)			
Determinism			
Formal definitions **(how many member of the tuple? What they are?)**			
Give state diagram for the language aa*			