8.2.1 - b) 000111

$q_0000111 \vdash Xq_100111 \vdash X0q_10111 \vdash X00q_1111 \vdash X0q_20Y11 \vdash Xq_200Y11 \vdash q_2X00Y11 \vdash Xq_000Y11 \vdash XXq_10Y11 \vdash$ $XX0q_1Y11 \vdash XX0Yq_111 \vdash XX0q_2YY1 \vdash XXq_20YY1 \vdash Xq_2X0YY1 \vdash XXq_00YY1 \vdash XXXq_1YY1 \vdash XXXYq_1Y1 \vdash$ $XXXYYq_11 \vdash XXXYq_2YY \vdash XXXq_2YYY \vdash XXq_2XYYY \vdash XXXq_0YYY \vdash XXXYq_3YY \vdash XXXYYq_3Y \vdash XXXYYYq_3B \vdash$ $XXXYYYBq_4B$ and the machine halts in accepting state q_4.

8.2.1 - c) 00111

$q_000111 \vdash Xq_10111 \vdash X0q_1111 \vdash Xq_20Y11 \vdash q_2X0Y11 \vdash Xq_00Y11 \vdash XXq_1Y11 \vdash XXYq_111 \vdash XXq_2YY1 \vdash Xq_2XYY1 \vdash$ $q_2XXYY1 \vdash Xq_0XYY1 \vdash XXq_0YY1 \vdash XXYq_3Y1 \vdash XXYYq_31$ and the machine halts in non-accepting state q_3.

8.2.2 - b) {anbncn | n >= 1}

This will be much like the machine given in figure 8.9.

State	a	b	c	X	Y	Z	B
q_0	(q_1,X,R)	–	–	–	(q_5,Y,R)	–	–
q_1	(q_1,a,R)	(q_2,Y,R)	–	–	(q_1,Y,R)	–	–
q_2	–	(q_2,b,R)	(q_3,Z,L)	–	–	(q_2,Z,R)	–
q_3	(q_3,a,L)	(q_3,b,L)	–	(q_0,X,R)	(q_3,Y,L)	(q_3,Z,L)	–
q_5	–	–	–	–	(q_5,Y,R)	(q_6,Z,R)	–
q_6	–	–	–	–	–	(q_6,Z,R)	(q_π,B,R)
q_π	–	–	–	–	–	–	–

The components of the Turing machine are $Q = \{q_0,q_1,q_2,q_3,q_5,q_6,q_\pi\}$, q_0 is the starting state, q_π is the accepting state, $\Sigma = \{a,b,c\}$, $\Gamma = \{a,b,c,X,Y,Z,B\}$, the blank is B, and δ is given above.

8.2.2 - c)

{wwR | is any string of 0's and 1's}

Here we need states that search for the left and right ends of the string, depending on whether a 0 or 1 was found at the beginning, then backtracking to see if a 0 or 1 is at the end.

State	0	1	B
q_0	(q_1,B,R)	(q_2,B,R)	(q_π,B,L)
q_1	$(q_1,0,R)$	$(q_1,1,R)$	(q_3,B,L)
q_2	$(q_2,0,D)$	$(q_2,1,D)$	(q_4,B,L)
q_3	(q_5,B,L)	–	–
q_4	–	(q_5,B,L)	–
q_5	$(q_5,0,L)$	$(q_5,1,L)$	(q_0,B,R)
q_π	–	–	–

The components of the Turing machine are $Q = \{q_0,q_1,q_2,q_3,q_5,q_\pi\}$, q_0 is the starting state, q_π is the accepting state, $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$, the blank is B, and δ is given above.

8.2.3 - a)

state \symb.	$	0	1	#	Explanation
q_0	$(q_R, \$, R)$	–	–	–	Initial $ reader
q_R	–	$(q_R, 0, R)$	$(q_R, 1, R)$	$(q_1, \#, L)$	Seeking end of string
q_1	$(q_f, 1, L)$	$(q_L, 1, L)$	$(q_1, 0, L)$	–	Recursive addition
q_L	$(q_f, \$, R)$	$(q_L, 0, L)$	$(q_L, 1, L)$	–	Moving left
q_f	–	–	–	–	

Note: in q_1 we branch according to current digit: if 0, make it 1 and go into "moving left" state; if 1, make it 0 and "carry one", i.e. move left and stay in the same state q_1 (like a recursive call); if $ then N was a string of 1's and we need to convert $ to 1 and finish.

8.2.3 - b)

Show the sequence of ID's of your TM when given input $111.

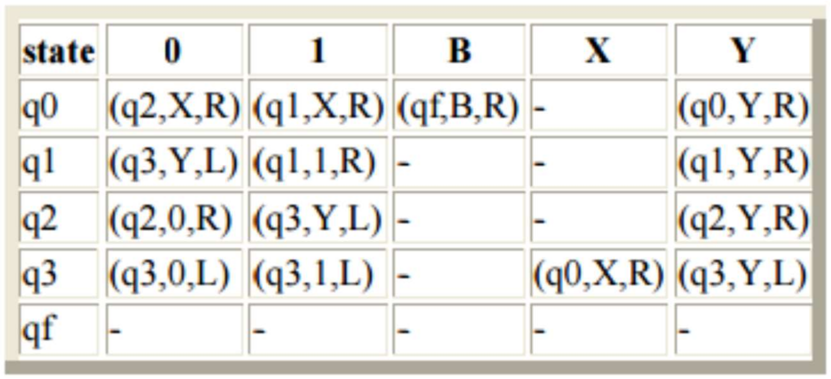
$$q_0\$111 \vdash \$q_R111 \vdash \$1q_R11 \vdash \$11q_R1 \vdash \$111q_R \vdash$$
$$\$11q_11 \vdash \$1q_110 \vdash \$q_1100 \vdash q_1\$000 \vdash q_B1000 \vdash q_f1000$$

8.2.5

This TM only moves right on its input. Moreover, it can only move right if it sees alternating 010101... on the input tape. Further, it alternates between states q0 and q1 and only accepts if it sees a blank in state q1. That in turn occurs if it has just seen 0 and moved right, so the input must end in a 0. That is, the language is that of regular expression (01)*0.

8.3.1

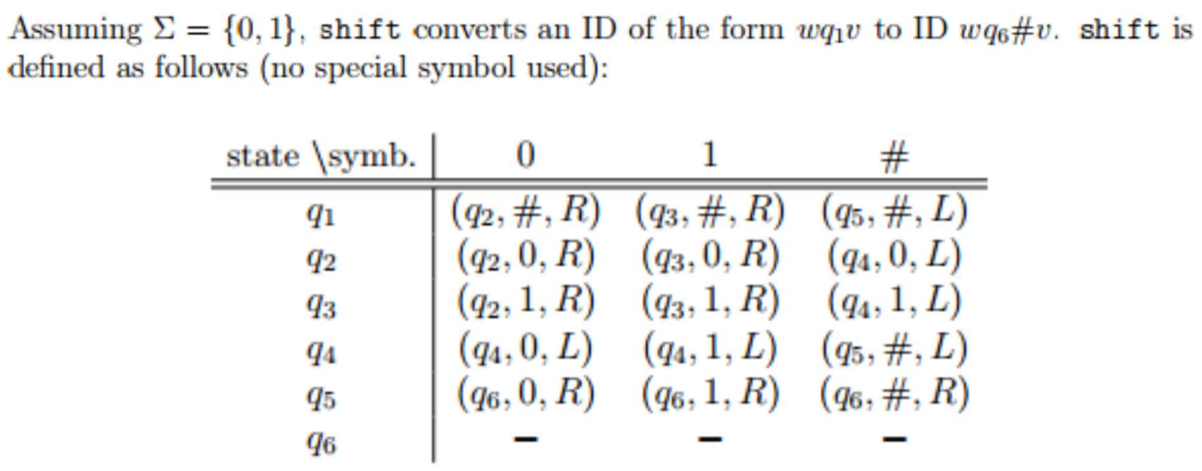
Here is the transition table for the TM:



state	0	1	B	X	Y
q0	(q2,X,R)	(q1,X,R)	(qf,B,R)	-	(q0,Y,R)
q1	(q3,Y,L)	(q1,1,R)	-	-	(q1,Y,R)
q2	(q2,0,R)	(q3,Y,L)	-	-	(q2,Y,R)
q3	(q3,0,L)	(q3,1,L)	-	(q0,X,R)	(q3,Y,L)
qf	-	-	-	-	-

In explanation, the TM makes repeated excursions back and forth along the tape. The symbols X and Y are used to replace 0's and 1's that have been cancelled one against another. The difference is that an X guarantees that there are no unmatched 0's and 1's to its left (so the head never moves left of an X), while a Y may have 0's or 1's to its left. Initially in state q0, the TM picks up a 0 or 1, remembering it in its state (q1 = found a 1; q2 = found a 0), and cancels what it found with an X. As an exception, if the TM sees the blank in state q0, then all 0's and 1's have matched, so the input is accepted by going to state qf. In state q1, the TM moves right, looking for a 0. If it finds it, the 0 is replaced by Y, and the TM enters state q3 to move left a look for an X. Similarly, state q2 looks for a 1 to match against a 0. In state q3, the TM moves left until it finds the rightmost X. At that point, it enters state q0 again, moving right over Y's until it finds a 0, 1, or blank, and the cycle begins again

8.3.2



Assuming $\Sigma = \{0, 1\}$, **shift** converts an ID of the form wq_1v to ID $wq_6\#v$. **shift** is defined as follows (no special symbol used):

state \symb.	0	1	#
q_1	$(q_2, \#, R)$	$(q_3, \#, R)$	$(q_5, \#, L)$
q_2	$(q_2, 0, R)$	$(q_3, 0, R)$	$(q_4, 0, L)$
q_3	$(q_2, 1, R)$	$(q_3, 1, R)$	$(q_4, 1, L)$
q_4	$(q_4, 0, L)$	$(q_4, 1, L)$	$(q_5, \#, L)$
q_5	$(q_6, 0, R)$	$(q_6, 1, R)$	$(q_6, \#, R)$
q_6	–	–	–

8.4.2 - b)

For the Input 011

For clarity, we put the state in square brackets below. Notice that in this example, we never branch. *[q_0]01 |- 1[q_0]1 |- 10[q_1] |- 10B[q_2]*

8.4.6

We design a 2-tape TM the following way

(a) Assume that the input is given as $\triangleright x$ on the first tape.

(b) Copy x in reverse onto the second tape, so that the second tape contains the string $\triangleright x^R$.

(c) Make appropriate transitions so that both tape heads are at the leftmost ends of their respective tapes.

(d) In this configuration, sweep from left to right on both strings till a blank $\sqcup$ is reached on both strings or it is discovered that x is not of the form $0^n \cdot 1^n, n \geq 0$.

The details of our Turing Machine $M = (Q, \Sigma, \delta, s)$ are as follows:

(i) $Q = \{s, q_0, q_1, q_2, q_3, \text{“}yes\text{''}, \text{“}no\text{''}\}$.

(ii) $\Sigma = \{0, 1, \sqcup, \triangleright\}$.

(iii) The transition function $\delta()$ is defined by the table below:
It is important to note that there should be a total of $6 \times 4 \times 4 = 96$ entries in the transition table. We impose the condition that for all the missing entries, the Turing Machine enters state “no'' and halts.

$q \in Q$	$\sigma_1 \in \Sigma$	$\sigma_2 \in \Sigma$	$\delta(q, \sigma_1, \sigma_2)$
s	$\triangleright$	$\sqcup$	$(s, \triangleright, \rightarrow, \sqcup, \rightarrow)$
s	0	$\sqcup$	$(s, 0, \rightarrow, \sqcup, \rightarrow)$
s	1	$\sqcup$	$(s, 1, \rightarrow, \sqcup, \rightarrow)$
s	$\sqcup$	$\sqcup$	$(q_0, \sqcup, \leftarrow, \sqcup, \leftarrow)$
q_0	0	$\sqcup$	$(q_0, 0, \leftarrow, \sqcup, -)$
q_0	1	$\sqcup$	$(q_0, 1, \leftarrow, \sqcup, -)$
q_0	$\triangleright$	$\sqcup$	$(q_1, \triangleright, \rightarrow, \sqcup, -)$
q_1	0	$\sqcup$	$(q_1, 0, \rightarrow, 0, \leftarrow)$
q_1	1	$\sqcup$	$(q_1, 1, \rightarrow, 1, \leftarrow)$
q_1	$\sqcup$	$\triangleright$	$(q_2, \sqcup, \leftarrow, \triangleright, -)$
q_2	0	$\triangleright$	$(q_2, 0, \leftarrow, \triangleright, -)$
q_2	1	$\triangleright$	$(q_2, 1, \leftarrow, \triangleright, -)$
q_2	$\triangleright$	$\triangleright$	$(q_3, \triangleright, \rightarrow, \triangleright, \rightarrow)$
q_3	$\sqcup$	$\sqcup$	$(\text{“}yes\text{''}, \sqcup, -, \sqcup, -)$
q_3	0	1	$(q_3, 0, \rightarrow, 1, \rightarrow)$
q_3	1	0	$(q_4, 1, \rightarrow, 0, \rightarrow)$
q_4	$\sqcup$	$\sqcup$	$(\text{“}yes\text{''}, \sqcup, -, \sqcup, -)$
q_4	1	0	$(q_4, 1, \rightarrow, 0, \rightarrow)$

8.4.7

For part 1, guess whether to move left or right, entering one of two different states, each responsible for moving in one direction. Each of these states proceeds in its direction, left or right, and if it sees it enters state p. Technically, the head has to move off the $, entering another state, and then move back to the, entering state p as it does so. For Part 2, doing the same thing deterministically, is trickier, since we might start off in the wrong direction and travel forever, never seeing the $. Thus, we have to oscillate, using left and right end markers X and Y, respectively, to mark how far we have traveled on a second track. Start moving one cell left and leave the X. Then, move two cells right and leave the Y. Repeatedly move left to the X, move the X one more cell left, go right to the Y, move it once cell right, and repeat. For part 3 we shall use at this time, we can move left or right to the other end marker, erase it, move back to the end where the $ was found, erase the other end marker, and wind up.

8.4.8

There would be 10 tracks. Five of the tracks hold one of the symbols from the tape alphabet, so there are 75 ways to select these tracks. The other five tracks hold either X or blank, so these tracks can be selected in 25 ways. The total number of symbols is thus 75 * 25 = 537,824. Exercise 8The number of symbols is the same. The five tracks with tape symbols can still be chosen in 75 ways. The sixth track has to tell which subset of the five tapes have their head at that position. There are 25 possible subsets, and therefore 32 symbols are needed for the 6th track.

8.4.9

Since any TM may be simulated on a k-head TM using only the first tape, it only remains to show that any k-head TM Mk may be simulated by a standard TM. For this we may instead show Mk may be simulated by a 3-tape TM M since we know the languages accepted by these are the same as those accepted by ordinary TM’s. Let M have an input tape containing the input to Mk, and 2 working tapes, the first of which begins blank. Let the symbols used by M on this working tape be the same as those used by Mk and let the set of symbols used on the second work tape be P ow([k]). When a symbol for a subset S of [k] = {1, . . . , k} appears in the i’th cell of the second work tape we will interpret this to mean the set S of heads are scanning the corresponding cell of the input tape. So, let the second work tape start with the symbol for {1, . . . , k} on its scanned cell and let all the rest of its cells be blank. Let the set of states of M be the Cartesian product: the set states of Mk × the set of additional states required as control states to execute the following algorithm (so M always knows both its Mk-state and its control state)

1. Apply the transition function of Mk to the current Mk-state and the vector of symbols scanned by the k emulated heads. Suppose this transition gives ((s1, d1) , . . . , (sk, dk)).
2. Go through the ((s1, d1) , . . . , (sk, dk)) in order (starting at i = 1 and going to i = k). Using the symbols written on all the cells of the second work tape as a guide (since they encode the location of the k simulated heads) write each symbol si on the first working tape in the current position of the i’th head.
3. Update the entries in the second work tape so that they reflect the new positions of the k heads (as determined by the di).
4. Now transfer each of the non-blank symbols on the first work tape to the input tape (in corresponding cell) and replace with blanks on the first work tape.
5. go to 1. If at any point during the execution of the algorithm Mk accepts then let M accept. Given Mk with input x and M as described (with corresponding input) it is easily verified that the input tape of M will always exactly match that of Mk after each time a new transition is called in both machines (and 1.-5. are executed by M). Since M accepts if and only if Mk does, this shows they accept the same language.

8.4.10

Let us and A language accepted by a standard Turing machine M1 can be accepted by a 2D TM M2. M2 can directly simulate all the steps of M (while using only one row of the grid). We will show that a language accepted by a 2D TM M2 can be accepted by a multi-tape Turing machine M1. Thus it will follow that it can be accepted by a standard single-tape TM. Let us number the cells of M2 by pairs of integers (x, y) where the initial position of M2's head is numbered (0, 0).

This splits the grid into four quadrants. M1 stores the pair of integers on the first "position" tape. M1 will have four one-way infinite "grid" tapes, each representing a quadrant of the 2D grid (e.g. the first tape represents the quadrant where $x \geq 0$, $y \geq 0$). To simulate an access to an element of the grid, M1 uses the signs of x, y to determine the tape on which the symbol is stored.

The absolute values of x, y are used to determine the position of the element on the grid tape, using the standard diagonal numbering function

$$f(a,b) = \frac{(a+b)(a+b+1)}{2} + b.$$

The value of f(|x|, |y|) is calculated on an additional "scratch" tape. The simulation of a single transition of M2 is then done as follows:

1. Use the signs of x and y to determine the grid tape which should be used.
2. Compute a = f(|x|, |y|) using the scratch tape.
3. Move the appropriate grid tape head to the a-th cell. (Note that grid tapes are one-way infinite, and we can use a special marker symbol to mark the leftmost cell.)
4. Based on the current state of M2 and the symbol read, determine the next state, overwrite the current cell on the grid tape, and simulate the move of the head of M2 by increasing or decreasing x or y on the position tape.