Jay Patel
CS 331
HW 9
Professor Oliver

Q1.
(b) {w|w contains twice as many 0s as 1s}

Implementation:
M = "on input w:

1. Scan w, if a '1' is found, replace with Z and move to head of the tape back to the left hand side and then goes to 2. If there's no '1' left and no '0' left, go to 4 else to go 5.
2. Scan w, if a '0' is found, replace with Z and go to 3. If there's no '0' left go to step 5.
3. Continue scan w, if another '0' is found, replace with Z and move the head of the tape to the left and then go to 1. If there's no '0' go to 5.
4. Accept
5. Reject

(c) {w|w does not contains twice as many 0s as 1s}

M = "on input w:

1. Scan the tape and mark the first 1, 1 that has not been marked. If no unmarked 1's are found go to 5 else move the head back to the start of the tape.
2. Scan the tape till an unmarked 0 is found, marks the 0 if no 0's are found accept.
3. Scan the tape once again until an unmarked 0, if no 0's are found accept.
4. Move the head back to the start of the tape and go to 1
5. Move the head back to the start of the tape, scan and see if any unmarked 0s are found. If none are found the reject else accept.

Q2.

Steps:

1. Let say that M is an ordinary TM. Let M' be a left reset machine.
2. If M makes a right transition then so does M'
3. If M makes a right transition then M' has to do a number of steps to copy a right transition. M' will make the position of the head. It will replace at its current position with something different like b*
4. When M' does a left reset it shifts each non-marked character to its right. When M' does a second left reset it travels with right transition until it reaches the spot that has been marked and then M' moves to make the same moves of M
5. Left transition of M can be called by equivalent operations in M'
6. For simulation of left reset in M, mark the left most at the beginning of the function.
7. Mark the left most character is reached when left reset is called and then M' can be functionally equivalent on M, and hence it will recognize the class of tuning recognizable languages.

Q3.

On a TM, we can easily create any DFA by stay set instead of left. In the modification of nontrivial upon reading a blank is to add transition.

Inside, state F to q_{accept}

Outside, state F to q_{reject}

So, M = (Q, Σ, Γ, δ, $q_0, q_{accept}, q_{reject}$)
In order to build DFA that identifies similar language is:
(Q', $\epsilon', \delta', {q_0}', F'$)

We know that M cannot move to left neither it can read on the tape as soon as it moves to right.

So,
Q' = Q, Σ' = Σ, $q_0' = \ q_0$ and the transition function:

$$\delta'(q,\delta) = \begin{cases} q & \text{if q}\epsilon(q_{accept,}q_{reject} \\ q_{reject} & \text{if m starts at state q} \\ q' & \text{when m moves to right} \end{cases}$$

Finally examine that there are finally many state alphabet, which ends up looping on it and moves to the right and hence δ' is well defined.
So, q ∈ $Q, q \ \neq \ q_{accept}, q_{reject}$

Q4.

(b) Concatenation: Suppose L1 and L2 are two decidable languages and M1 and M2 are two Turing Machines that decides L1 and L2. Let M3 decide L1.L2 for string w.
M3 = “on input w:

1. Copy w on 2^{nd} tape and reset all heads to the front of the tapes
2. Run M1 on w. If M1 accepts w, run M2 on w
3. If M2 accepts 2 accepts, else rejects

This means M3 accepts:
w1 if M1 accepts w, then M2 accepts w.

Both M1 and M2 are deciders because M3 is a decider and it will be a finite number of steps to decide whether to accept or reject w.

(c) Star: Suppose L is the decidable language and ML is the TM, which decides L and L* be decidable by TM ML*
For string w:
ML* = “On input w:

1. Split w into various substring w1, w2, w3…wn
2. Run ML on each of w1, w2..wn. It accepts only if ML accepts all. Reject if ML halts and rejects for any i.

This tells us, that ML* is a decider because ML is a decider and it will be finite number of steps if we can split w into n substring w1,w2…wn which can be accepted by ML

(d) Complementation: Suppose L is decidable language and ML is a TM, which decides L. L’ be decidable by TM ML’
Then for string w: ML’ = “on input w:

1. Run ML on w. if it accepts, reject, otherwise reject

ML’ accepts if ML rejects the input w
Since it takes finite number of steps to determine where the input is rejected or accepted we can tell that ML’ is a decider since ML is a decider.

(e) Intersection: Supper L1, L2 are two decidable languages and M1 and M2 be two TM, which decide L1 and L2 respectively. M3 decide L1 intersection L2. Then for string w
M3 = “on string w:

1. Run M1 on w. If rejects, reject
2. Run M2 on w. if rejects, reject, otherwise, Accept

This means M3 accepts w1 if M1 accepts w and M2 accepts w

It will be finite number of steps to decider whether to accept or reject w since M3 is a decider because both m1 and m2 are deciders.

Q5.

(b) Concatenation: Lets say that M1 and M2 are the TM that recognizes language L1 and L2. In order have a language that recognizes concatenation for input x we need to construct a language M3 one must, cut x into substrings wy non-deterministically. Then run 2 on TM M1. Then check if it accepts then run y on TM M2. M2 accepts, then it accepts x.

(c) Star: In order to construct M3 that accepts L1 one must check if the input of x is empty, if it is then it accepts else as above cut the string into multiple non-deterministically. Later, on each substring we run TM M1, if the substring of TM M1 accepts these substrings then we accept. There are a finite number of steps of M3 if x can be cut into finite number of substrings.

(d) Intersection: if both of the machine M1 and M2 on input x are in parallel accepts then the intersection of L1 and L2 is defined and they all accept.

(e) Homomorphism: Assume that L is a Turing recognizable. Lets say that w is the input of M, for the second tape M will go through all the strings of x over the L and checks that F (x) = w, we know that for every character in x will be the same as f (x) if its accepted. So, if x is accepted, M accepts w.