Q5 Transitions Solution

C1. (a) (S) f 1 q 0

(b) (S) i 2 r S c 0

(c) (S) o 2 r S h 0

(d)(S) n 1 q 0

C2. Put a tick (✓) in the box if the network generates this string,

(a)	(b)	(c)	(d)	(e)	(f)	(g)
	✓		✓	✓		

C3. Change rule h as follows: S: eight → 0

Additional rule S : eigh → 1,2
(Some other fixes are possible, but this is the most elegant and obvious)

Explanation

Transition networks are a very simple tool in computational linguistics and simultaneously serve two purposes. One, seen here, is to capture the way a text string can be 'generated', i.e. start from S and follow the rules. But the network can also be used for 'parsing' or analyzing a string to see if it's legal. Actually that is what C2 asked *you* to do. What you did is probably similar to what a computer program would do. For example, to analyse 'sixteen', look for a rule that starts from state S and generates any of the first few characters: yes, [f] generates 'six'. Rule [f] says you can end (0) or go to state 1 or 2. Which rules can start from either of those states? Rules [q] nd [r]. Do either of those rules lead you on to the next part of the string, 'teen'? Yes, rule [q]. And, crucially, does that rule then lead you to the exit state? Yes.

Let's try the same thing with C2e 'fortythirty'. We want the network to fail for this. Let's see if it does. First we look for a rule that starts from state S and generates any of the first few characters: only rule [p] works. Note that rule [d] requires 'four', not 'for'. Rule [p] can only lead to one state, 2. And only one rule, [r] can lead off from that state. Rule [r] is looking for 'ty', which we do have. So far so good. Where can you go from rule [r]? Back to the start S, or finish (0). Can we generate 'thirty' from S? Yes we can (you should really trace the steps, but they are similar to 'forty': rule [m] then rule [r]. So 'fortythirty' is legal, even though we don't want it to be.