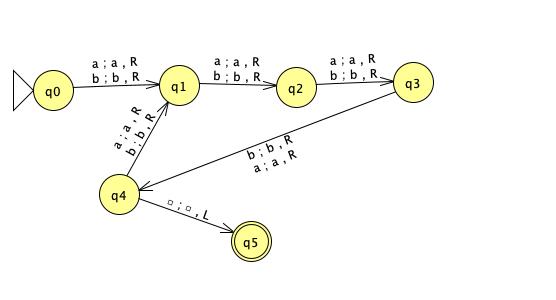
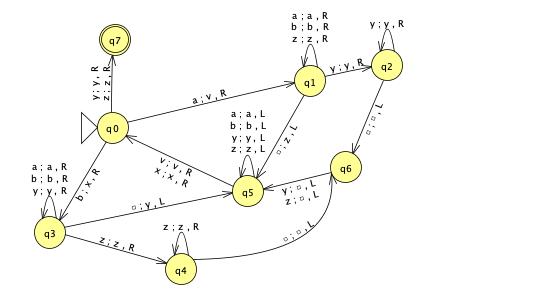
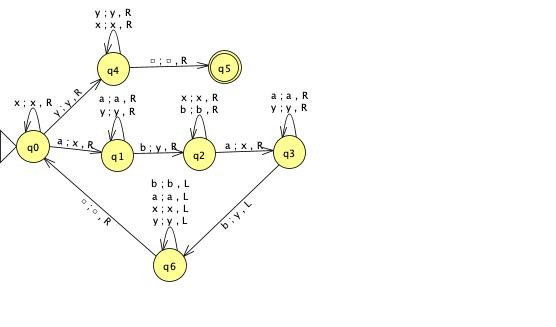
CSci 435: Formal Languages and Automata

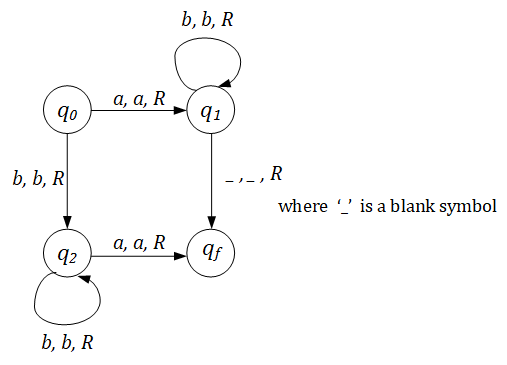
Instructor: Dr. M. E. Kim Name: \_\_\_\_\_Derek Trom\_\_\_\_\_\_

**Home Assignment 7: 117/120 points + 25 points (optional)**

Q1. [20/20] For a given language below, construct a TM with a *single final state* that accepts it.

1. [6/6] L = {w ||*w*|is a multiple of 4} where Σ = {*a*, *b*}.
   1. 
2. [7/7] L = {w | *na*(*w*) ≠ *nb*(*w*)} where Σ = {*a*, *b*}.
   1. 
3. [7/7] L = {w | *anbn anbn* | *n* ≠ 0} where Σ = {*a, b*}.
   1. 

Q2. [10] What language is accepted by the Turning machine whose transition graph is in the figure?

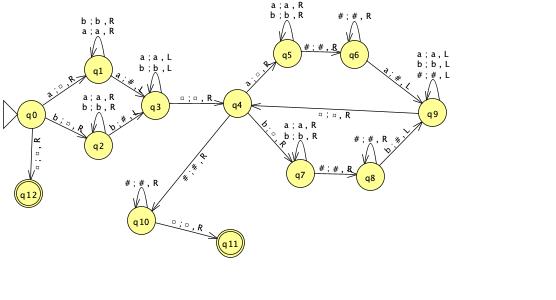


r = (bb\*a) + (ab\*) L = { bna | n > 0} ⋃ { abn | n ≥ 0 }

Q3. [8/10] Construct a TM that accepts L = {ww | w ∈ {*a, b*}+ }. Pg 240

Hint: This is a standard deterministic TM.

So, TM has to **find** the middle of the string first; then, compare two halves.



First, find the middle of the string; then compare the two halves.

1. Find the leftmost symbol of the input which is neither A nor B:

Mark the symbol by A if it is an *a* and replace it by B if it is b:

1. Move the read-write head to right and the rightmost symbol which is neither A nor B;

and mark it with A or B accordingly

1. Move the read-write head to the left, and repeat step 1 and step 2.

It must terminate in either step 1 or step 2 when all the input symbols has been replaced by A and B.

If the input is an even length string, the process will terminate in step 1 and find the middle of the string with the read-write stops at the starting of the second half of the input. Otherwise, the process will halt in step 2 which means the input is an odd length and is not in L.

Repeat

1. Find the leftmost symbol x ≠A/B.

If x = *a* then replace it by A;

If x = b then replace it by B.

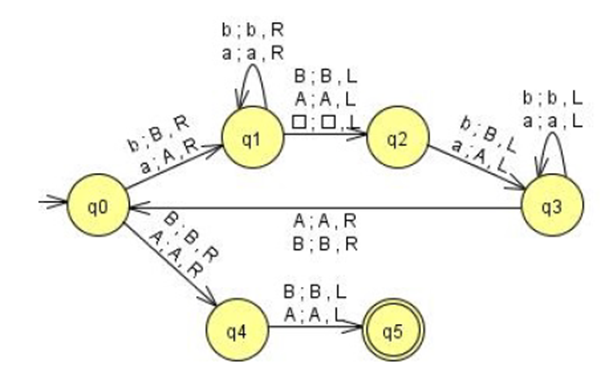
1. Move head to right and fine the rightmost symbol y ≠A/B.

If y = *a* then replace it by A;

If y = b then replace it by B.

1. Move head to left.

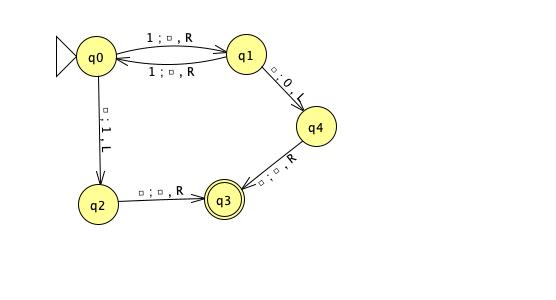
Until all symbols = A/B (i.e. all symbols are replaced by A or B)



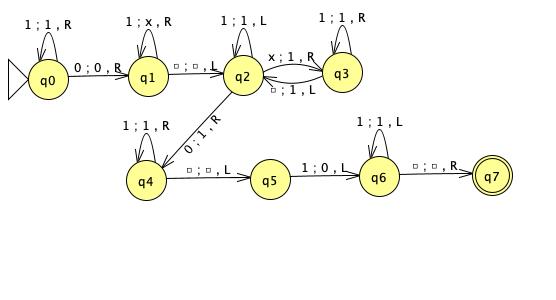
Q4. [20/20] Construct a TM that computes the following function

1. [10/10] .

The input *w* is in the unary representation.

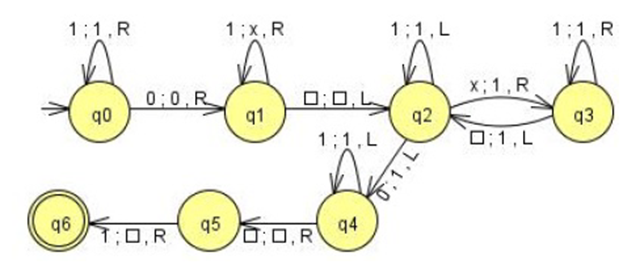


1. [10/10]  *f*(*x, y*) = *x* + 2*y.*



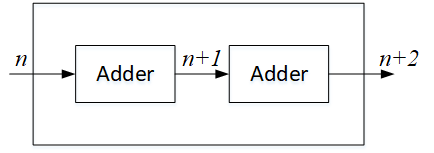
For a given problem w(x)0w(y), use the construction in Example 9.10 to duplicate w(y),

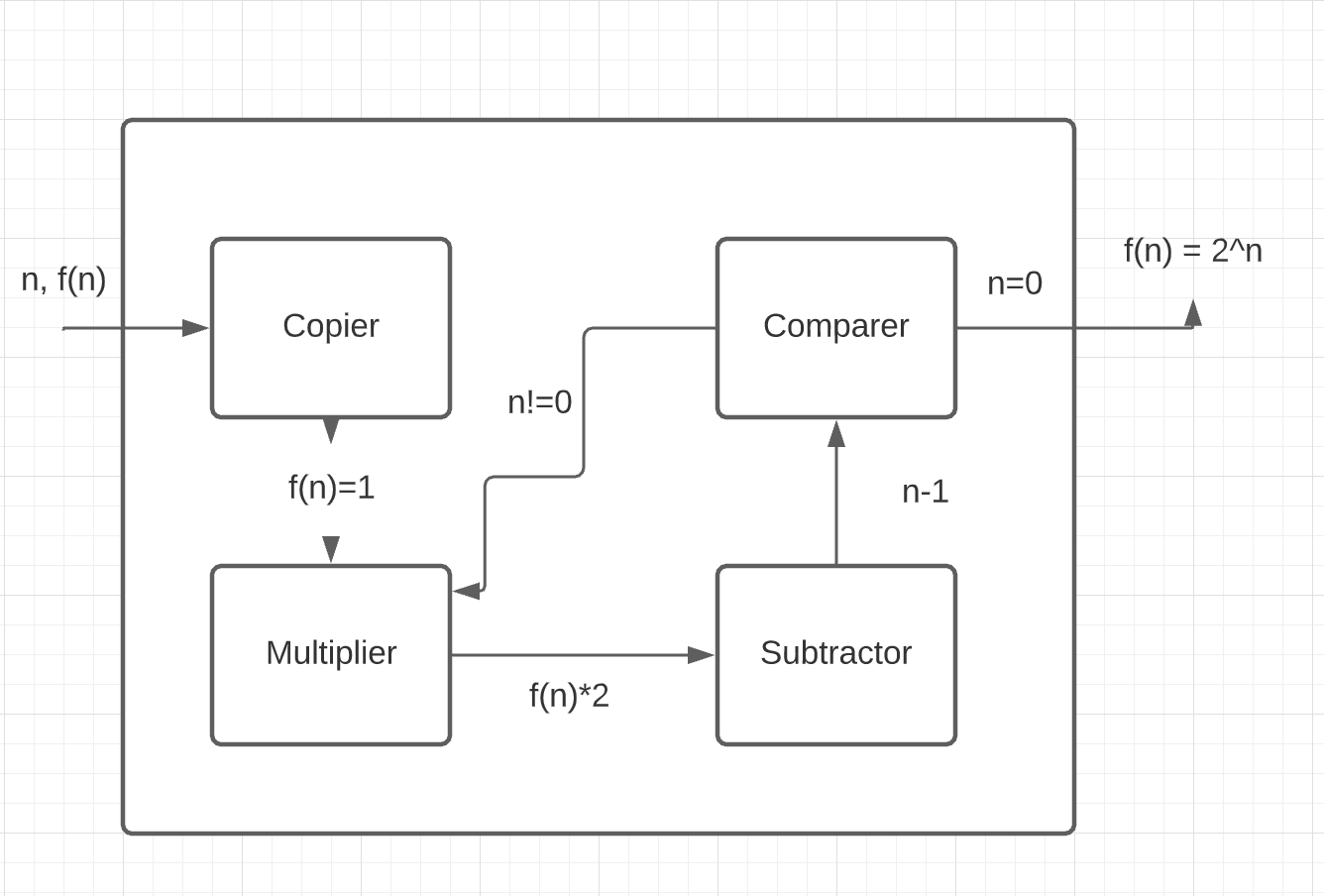
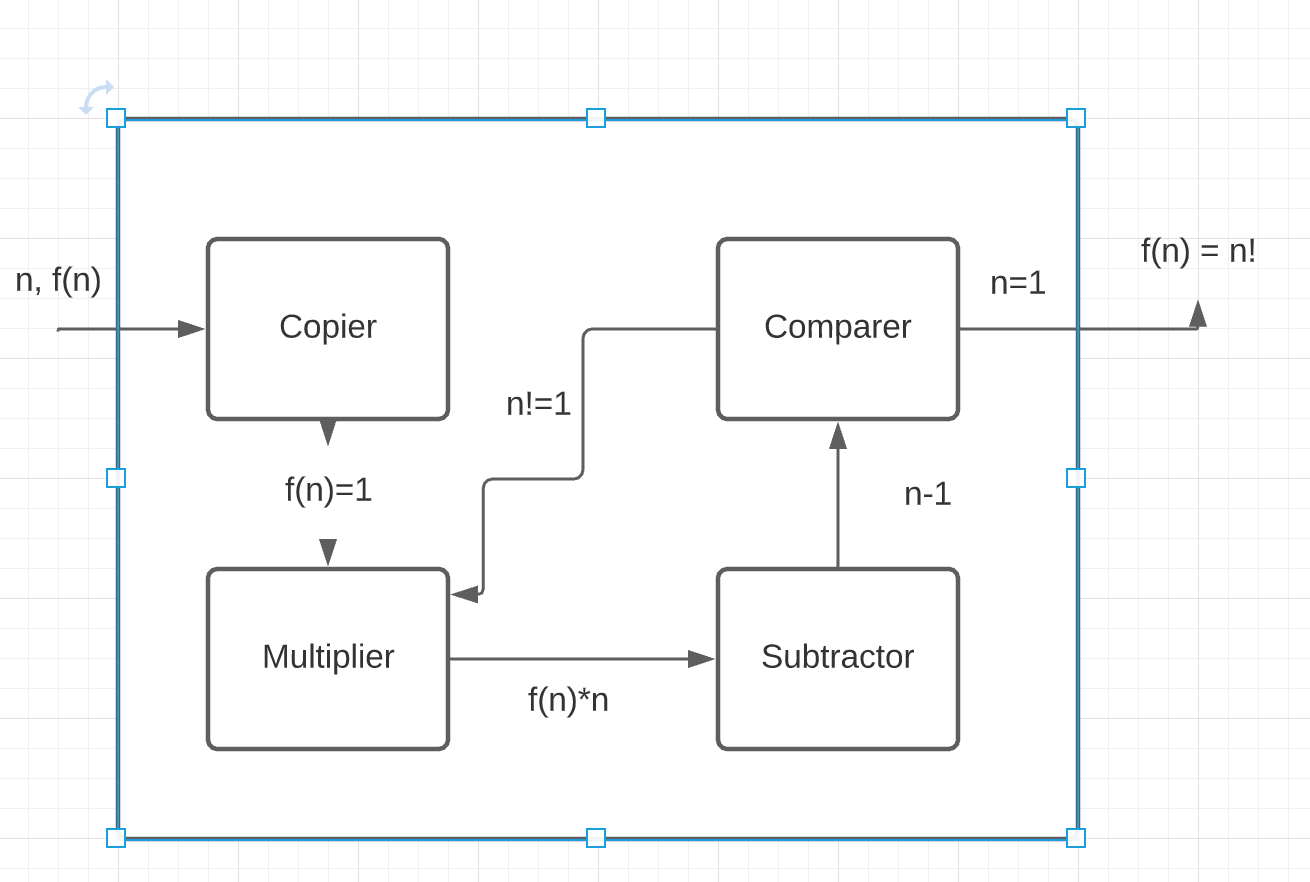
then add to w(x) to get the resulting string w(x)w(y)w(y).



Q5. [20/20] Using adders, subtractors, comparers, copiers or multipliers, draw block diagram for TM that compute the functions:

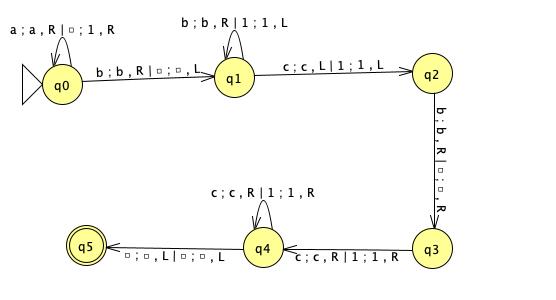
e.g.) *f*(*n*) = *n* + 2



1. [10/10]*f*(*n*) = *2n.*
   1. 
2. [10/10] *f(n) = n!*
   1. 

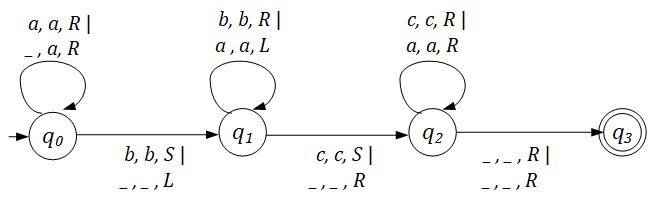
Q6. [13/15] For a two-tape Turing Machine,

1. [5/5] Give a *formal definition* of a transition function δ in two-tape TM.
   1. The multi-tape transition function is defined as δ: Q ⨯ Γn → Q ⨯ Γn ⨯ {L,R}n where n is the number of tapes, Q is the set of internal states, Γ is the tape alphabet, and L and R represent the move symbols. If we set n=2 we get δ(qn, x, y) = (qm, b, c, {L,R}, {L,R}) where x and y are the symbols read from the two tapes. The symbols that are written to the tapes are b and c respectively, with {L,R} being the directions for the tape to move.
2. [8/10] Construct a two-tape TM that accepts L = { *anbn cn* | *n* ≥ 1}



For a transition, δ (*qi, a, b*) = (*qj, c, d, L, R*),

let’s represent it by *a, c, L* | *b, d, R* for an edge from *qi,* to *q,.*



Q7.[0/15, optional] Construct a **Nondeterministic** TM (NTM) that accepts L ={ *wwRw* | *w* ∈ {*a, b*}+ }.

1. Draw its transition graph, (B) explain how your transitions work out and (C) how the nondeterministic simplifies the case.

Note that the middle of the string in wwR can be guessed in NTM.

Q8. [6/10] Give the encoding, using the suggested method in the slide of Chap.9-#25-#27, for

δ(*q1, a1*) = (*q1, a1*, R); δ(*q1, a2*) = (*q3, a1*, L); δ(*q3, a1*) = (*q2, a2*, L)

1010101011 00 101101110101 00 1110101101101

δ(*q1, a1*) = (*q1, a1*, R); 010101010110

δ(*q1, a2*) = (*q3, a1*, L); 01011011101010

δ(*q3, a1*) = (*q2, a2*, L): 011101011011010

Q9. [5/5] If *a* is encoded as 1, *b* as 11, R as 1, L as 11, decode the string 011010111011010.

δ(*q2, a*) = (*q3, b*, R);

Q10. [8/10, optional] Describe an algorithm that examines a string in {0, 1}+ to determine whether or not it represents an encoded Turing Machine.

Assume that each encoded transition function consists of 10 sets of 1’s separated by zeros. We can check each set of 10 1’s for each transition function to make sure the last set of ones is 1 or 11. This determines if the last sets contains L and R. We can verify number of 1’s in the string by dividing the string by 10. These two tests would show that it is most probably an encoded TM.

To determine if a given string in {0, 1}+ represents a transition function δ(*qi , ak*) = (*qj , al ,* L or R}) for an encoded Turning machine, we check that the input is of the form 0*x1*0*x2*0*x3*0*x4*0*x5*0, where the *xi* are strings of 1's, with the restriction that *x5* = 1 or 11.

Q11. [7/10] Describe how Linear Bounded Automata could be constructed to accept

L = { *an* | *n* is a prime number}.

Let us set up a two track LBA where track 1 contains an arbitrary number of a’s. Track 2 will contain the current divisor of the string starting with 2 a’s. Then we will divide by 3,4,5…and so on. If any division attempt succeeds this means that the string is not prime and will not be accepted. However if all divisions fail this would mean that it would be accepted.

One way to solve it is to divide the number of input *a*'s which is say *n* successively by 2, 3, … , ⎣*n*/2⎦ .

If the remainder of every division is nonzero, then *n* is a prime number and we accept the input. Otherwise, we reject the input.

To design a LBA, we use a 2-track machine as shown below.

The 1st –track contains the input, a string of *a*’s, and

the 2nd-track is the current divisor used to divide the number of *a*'s on the 1st-track.

We find the remainder for the division, say, by replace all the symbols by blanks on the 1st - track except those at the multiples of the divisor. Thus, if the rightmost symbol a is replaced by a blank after finishing the division, then we have a nonzero remainder. We continue the process of division, by increment the divisor by one, until either we accept or reject the input. The divisor starts with two *a*'s with a marker x on the track ⎣*n*/2⎦ from the leftmost *a* (i.e. a middle point), that can be done by matching symbols from outside towards the middle on 2nd -track.

The marker x is to denote the number corresponding to ⎣*n*/2⎦ +1.

