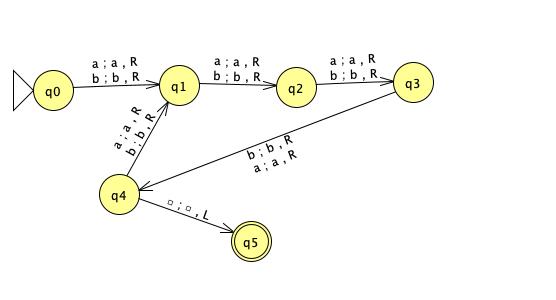
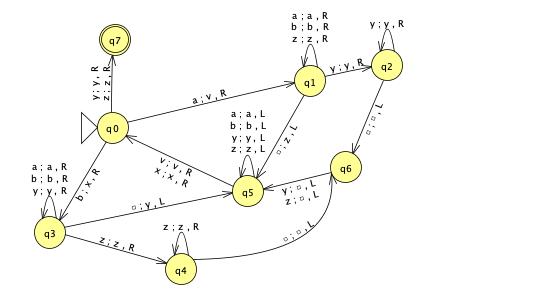
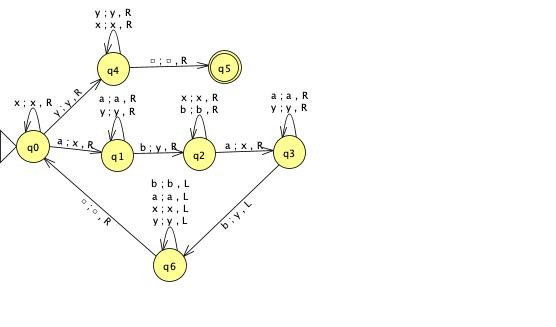
CSci 435: Formal Languages and Automata

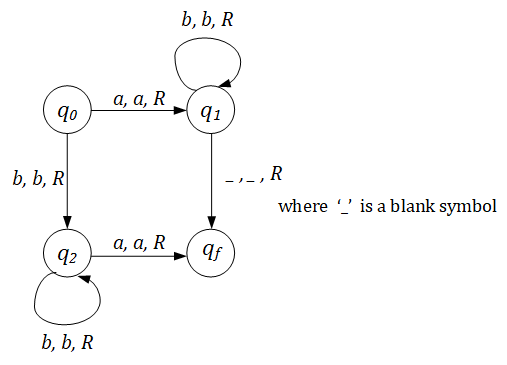
Instructor: Dr. M. E. Kim Name: \_\_\_\_\_Derek Trom\_\_\_\_\_\_

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

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

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

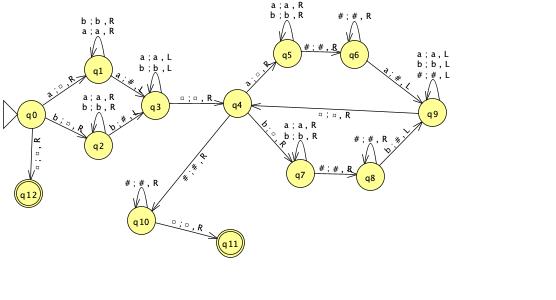


r = (bb\*a) + (ab\*) L = { bna | n > 0} ⋃ { abn | n ≥ 0 }

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

Hint: This is a standard deterministic TM.

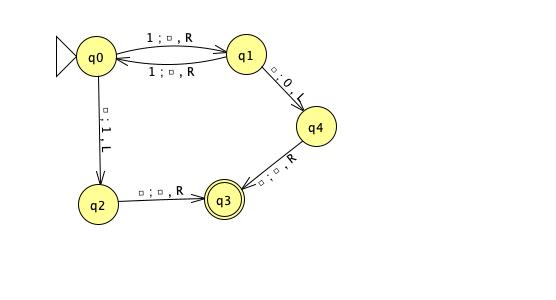
So, TM has to **find** the middle of the string first; then, compare two halves.

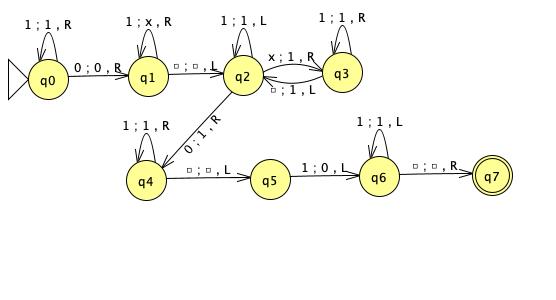


Q4. [20] Construct a TM that computes the following function

1. [10] .

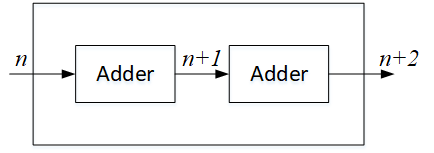
The input *w* is in the unary representation.

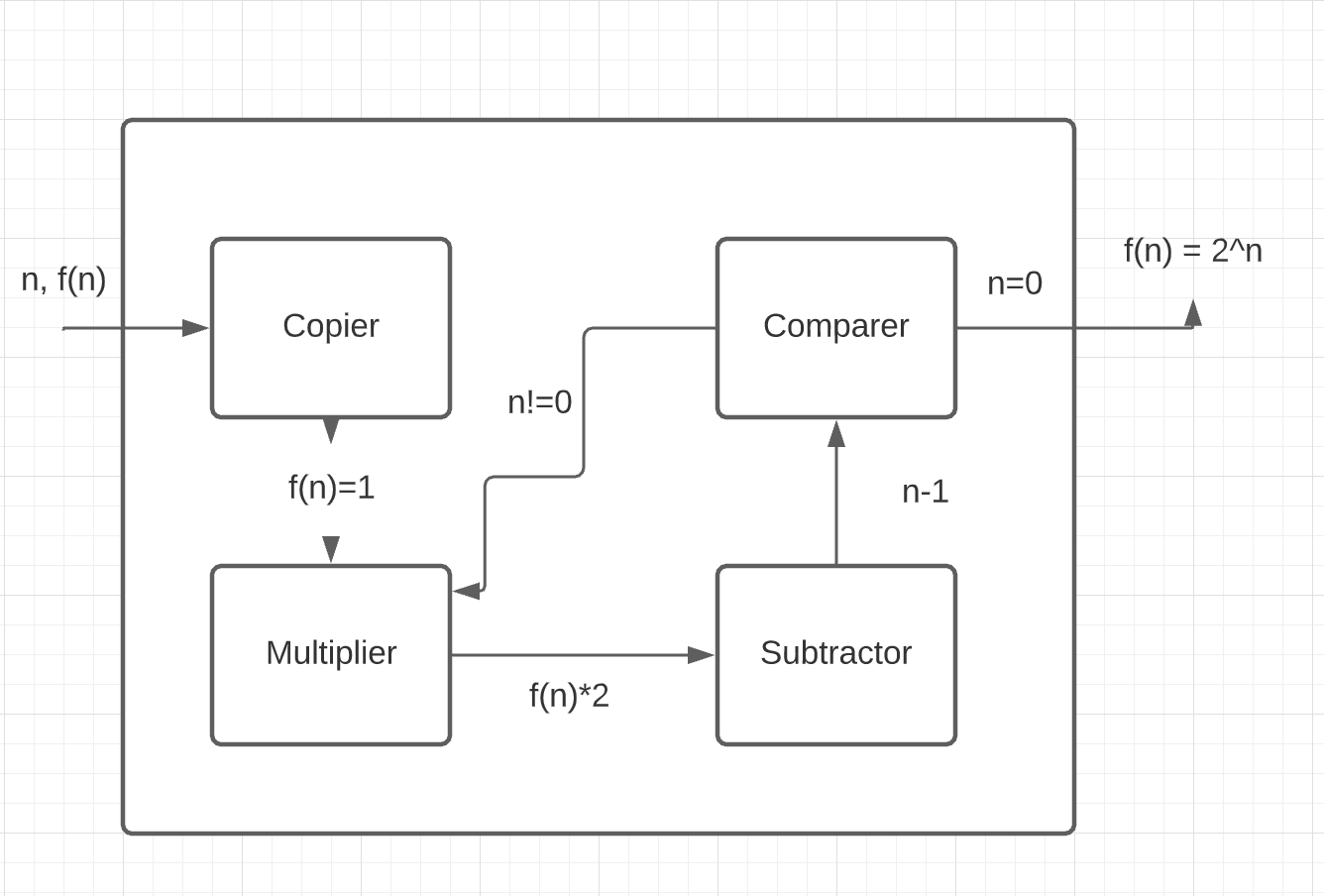
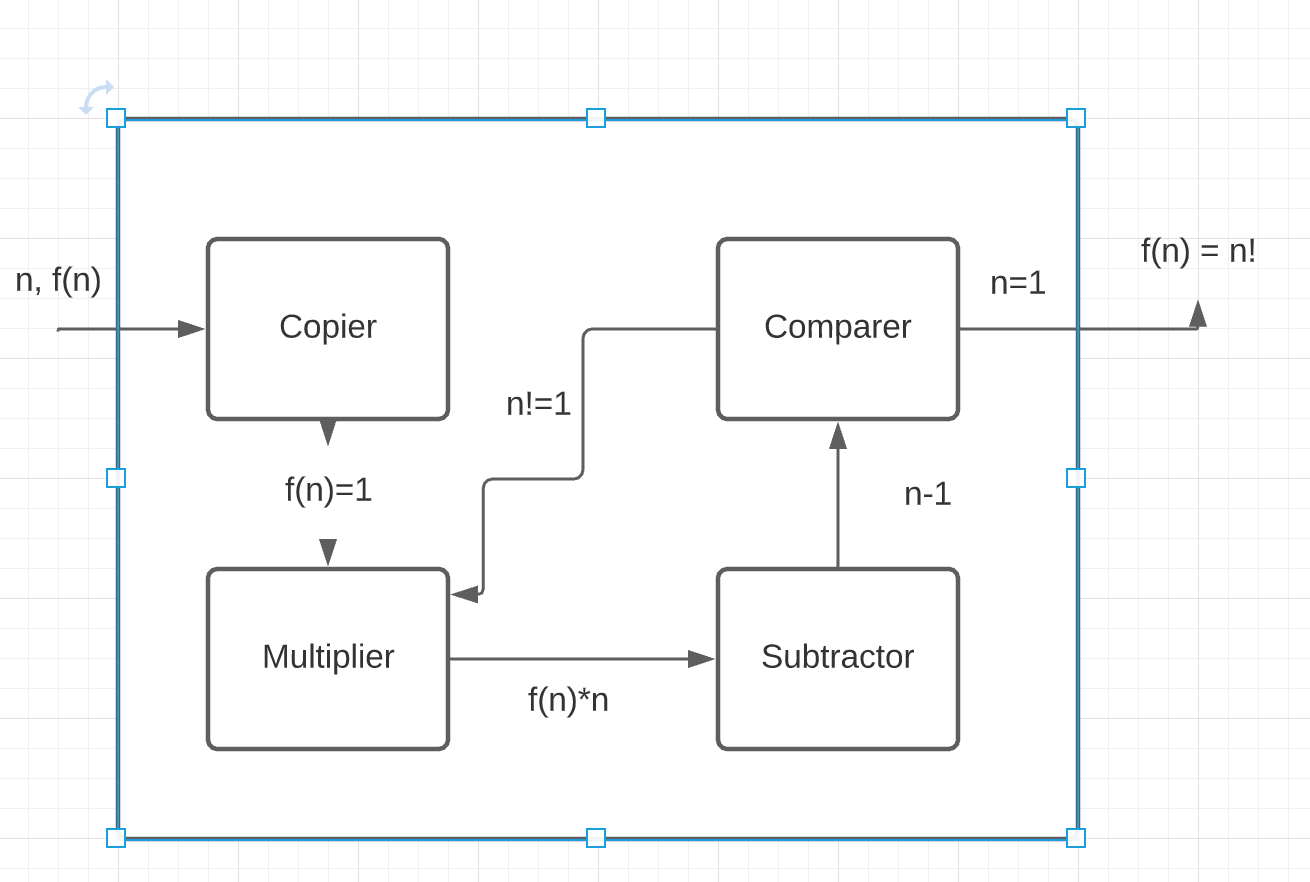


1. [10]  *f*(*x, y*) = *x* + 2*y.*
   1. 

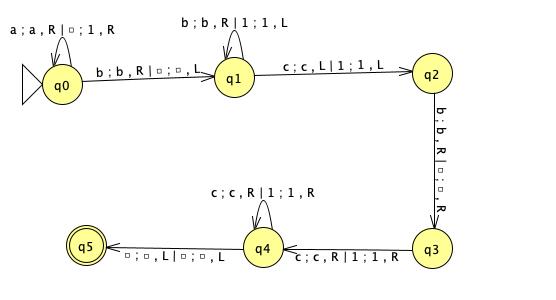
Q5. [20] Using adders, subtractors, comparers, copiers or multipliers, draw block diagram for TM that compute the functions:

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



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

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

1. [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. [10] Construct a two-tape TM that accepts L = { *anbn cn* | *n* ≥ 1}
   1. 

Q7.[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. [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

Q9. [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. [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.

Q11. [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.