7. Construct Turing Machines that will accept the following languages on $\{a, b\}$.

   (a) $L = L(aba^*b)$.

   Solution: q_0 is the initial state and q_f is the only final state. The read/write head is assumed to start on the first letter of the word.

$$\begin{aligned}
\delta(q_0, a) &= (q_1, a, R) \\
\delta(q_1, b) &= (q_2, b, R) \\
\delta(q_2, a) &= (q_2, a, R) \\
\delta(q_2, b) &= (q_f, b, R)
\end{aligned}$$

   (c) $L = \{w \mid |w| \text{ is a multiple of } 3\}$.

   Solution: q_0 is the initial state and the only final state. The read/write head is assumed to start on the first letter of the word.

$$\begin{aligned}
\delta(q_0, a) &= (q_1, a, R) \\
\delta(q_0, b) &= (q_1, b, R) \\
\delta(q_1, a) &= (q_2, a, R) \\
\delta(q_1, b) &= (q_2, b, R) \\
\delta(q_2, a) &= (q_0, a, R) \\
\delta(q_2, b) &= (q_0, b, R)
\end{aligned}$$

   (e) $L = \{w \mid n_a(w) = n_b(w)\}$.

   Solution: q_0 is the initial state and q_f is the only final state. This can be done in fewer transitions but the one below might make more sense. The read/write head is assumed to start on the first letter of the word. Starting with q_0 we mark the a as x and b as y and then shift into a state that will search for the corresponding letter. The state q_a searches for an a and the state q_b searches for a b, when found the machine returns to the first a or b in the string. The state q_r returns to the first character, the state q_s moves to the first a, b, or space, and the state q_t is there only to undo the needed left move at the end of q_s. At this point the process starts over again. If we hit a space while in q_a or q_b then there are an unequal number of a's and b's and hence we move into the state q_n which is not final and halts the machine. If we hit a space on q_0 then tere were an equal number of a's and b's and hence we move into the state q_f which is final and halts the machine.

$$\begin{aligned}
\delta(q_0, a) &= (q_b, x, R) & \qquad \delta(q_r, a) &= (q_r, a, L) \\
\delta(q_0, b) &= (q_a, y, R) & \qquad \delta(q_r, b) &= (q_r, b, L) \\
\delta(q_0, \Box) &= (q_f, \Box, L) & \qquad \delta(q_r, x) &= (q_r, x, L) \\
\delta(q_b, a) &= (q_b, a, R) & \qquad \delta(q_r, y) &= (q_r, y, L) \\
\delta(q_b, x) &= (q_b, x, R) & \qquad \delta(q_r, \Box) &= (q_s, \Box, R) \\
\delta(q_b, y) &= (q_b, y, R) & \qquad \delta(q_s, a) &= (q_t, a, L) \\
\delta(q_b, b) &= (q_r, y, L) & \qquad \delta(q_s, b) &= (q_t, b, L) \\
\delta(q_b, \Box) &= (q_n, \Box, R) & \qquad \delta(q_s, \Box) &= (q_t, \Box, L) \\
\delta(q_a, b) &= (q_a, b, R) & \qquad \delta(q_s, x) &= (q_s, x, R) \\
\delta(q_a, x) &= (q_a, x, R) & \qquad \delta(q_s, y) &= (q_s, y, R) \\
\delta(q_a, y) &= (q_a, y, R) & \qquad \delta(q_t, x) &= (q_0, x, R) \\
\delta(q_a, a) &= (q_r, x, L) & \qquad \delta(q_t, y) &= (q_0, y, R) \\
\delta(q_a, \Box) &= (q_n, \Box, R)
\end{aligned}$$

9. Construct a Turing Machine to compute the function $f(w) = w^R$, where $w \in \{0,1\}^+$.

   Solution: q_B is the initial state and q_f is the only final state. The read/write head is assumed to start on the first letter of the word. Starting with q_B we replace the 0 or 1 with an x and shift to a state that will move to the beginning of the word and write the memorized character. Then the machine shifts to the state, q_r that will move the read/write head to the right until an x is read, at which point the state shifts back to q_B, which moves right over all the x's and repeats the process when a 0 or 1 is read. If the q_B state reads a space the machine will shift to state q_E which will erase all of the x's and finally halt.

$$\begin{array}{rclrcl}
\delta(q_B,0) &=& (q_0,x,L) & \delta(q_0,0) &=& (q_0,0,L)\\
\delta(q_B,1) &=& (q_1,x,L) & \delta(q_0,1) &=& (q_0,1,L)\\
\delta(q_B,x) &=& (q_B,x,R) & \delta(q_0,\Box) &=& (q_r,0,R)\\
\delta(q_B,\Box) &=& (q_E,\Box,L) & \delta(q_r,0) &=& (q_r,0,R)\\
\delta(q_1,x) &=& (q_1,x,L) & \delta(q_r,1) &=& (q_r,1,R)\\
\delta(q_1,0) &=& (q_1,0,L) & \delta(q_r,x) &=& (q_B,x,R)\\
\delta(q_1,1) &=& (q_1,1,L) & \delta(q_E,x) &=& (q_E,\Box,L)\\
\delta(q_1,\Box) &=& (q_r,1,R) & \delta(q_E,0) &=& (q_f,0,L)\\
\delta(q_0,x) &=& (q_0,x,L) & \delta(q_E,1) &=& (q_f,1,L)
\end{array}$$

11. Design Turing Machines to compute the following functions for x a positive integer represented in unary.

    (a) $f(x) = 3x$.

    Solution: q_0 is the initial state and q_f is the only final state. The read/write head is assumed to start on the first letter of the word. Starting with q_0 we replace all of the 1's with x's. Then the machine will find the last x, change it to a 1, move to the end of the word and write two more 1's. The machine then moves to the left until it encounters an x, at which point the process repeats. If a space in encountered before an x the process is finished and the machine halts.

$$\begin{array}{rclrcl}
\delta(q_0,1) &=& (q_0,x,R) & \delta(q_3,\Box) &=& (q_r,1,L)\\
\delta(q_0,\Box) &=& (q_1,\Box,L) & \delta(q_r,1) &=& (q_r,1,L)\\
\delta(q_1,x) &=& (q_2,1,R) & \delta(q_r,x) &=& (q_s,x,R)\\
\delta(q_2,1) &=& (q_2,1,R) & \delta(q_s,1) &=& (q_1,1,L)\\
\delta(q_2,\Box) &=& (q_3,1,R) & \delta(q_r,\Box) &=& (q_f,\Box,R)
\end{array}$$

    (e) $f(x) = x \pmod 5$.

    Solution: q_0 is the initial state and q_f is the only final state. The read/write head is assumed to start on the first letter of the word. Starting with q_0 we erase the word using 5 different states to track the modulus. Once we hit a space the state we are in tells the machine how many 1's to rewrite, and hence it goes into a sequence of states to write the appropriate number of 1's back to the tape before it halts.

$$\begin{array}{rclrclrcl}
\delta(q_0,1) &=& (q_1,\Box,R) & \delta(q_0,\Box) &=& (q_{M_0},\Box,L) & \delta(q_{M_0},\Box) &=& (q_f,0,L)\\
\delta(q_1,1) &=& (q_2,\Box,R) & \delta(q_1,\Box) &=& (q_{M_1},\Box,L) & \delta(q_{M_1},\Box) &=& (q_f,1,L)\\
\delta(q_2,1) &=& (q_3,\Box,R) & \delta(q_2,\Box) &=& (q_{M_2},\Box,L) & \delta(q_{M_2},\Box) &=& (q_{M_1},1,L)\\
\delta(q_3,1) &=& (q_4,\Box,R) & \delta(q_3,\Box) &=& (q_{M_3},\Box,L) & \delta(q_{M_3},\Box) &=& (q_{M_2},1,L)\\
\delta(q_4,1) &=& (q_0,\Box,R) & \delta(q_4,\Box) &=& (q_{M_4},\Box,L) & \delta(q_{M_4},\Box) &=& (q_{M_3},1,L)
\end{array}$$

Use the primitives R, L, R_a, L_a, R_b, L_b, $R_\square$, $L_\square$, R_0, L_0, R_1, L_1, $R_{\overline{a}}$, $L_{\overline{a}}$, $R_{\overline{b}}$, $L_{\overline{b}}$, $R_{\overline{\square}}$, $L_{\overline{\square}}$, $R_{\overline{0}}$, $L_{\overline{0}}$, $R_{\overline{1}}$, $L_{\overline{1}}$, a, b, 0, 1, $\square$, A, S, Cp, Shl, Shr, N_L, N_R, W_E and W_B, and the tape alphabet of $\{a, b, 0, 1, \square\}$ where,

A — Adds one in binary, the read/write head begins and ends on the leftmost digit. So applying it to $\underline{1}00101$ produces $\underline{1}00110$. Also the number grows to the left, so $\square\underline{1}11$ produces $\underline{1}000$.

S — Subtracts one in binary, the read/write head begins and ends on the leftmost digit. So applying it to $\underline{1}00110$ produces $\underline{1}00101$. Also the number shrinks on the left, so $\underline{1}000$ produces $\square\underline{1}11$.

Cp — Copies a word. So $\underline{a}ba\square$ produces $\underline{a}ba\square aba$.

Shl — Shifts a word one space to the left. So $\square\underline{a}ba$ produces $\underline{a}ba\square$.

Shr — Shifts a word one space to the right. So $\underline{a}ba\square$ produces $\square\underline{a}ba$.

N_L — Moves the read/write head to the beginning of the next word to the left.

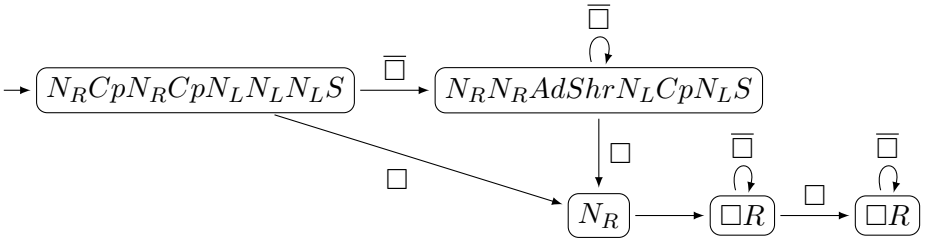
N_R — Moves the read/write head to the beginning of the next word to the right.

W_E — Moves the read/write head to the end of the word. If the read/write head is on a space the head does not move.

W_B — Moves the read/write head to the beginning of the word. If the read/write head is on a space the head does not move.

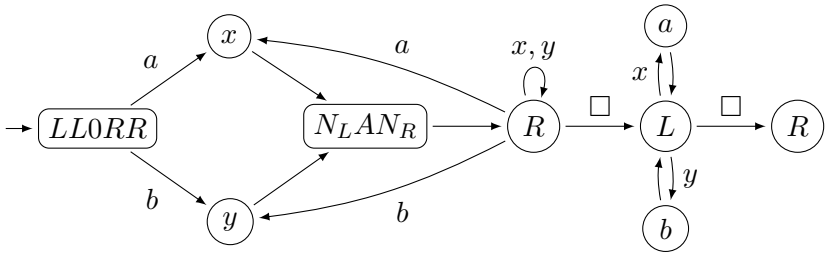
1. Construct a Turing machine, in diagram form, that takes two input numbers on the tape, n_1 and n_2, in binary form, and returns the binary form of $n_1 n_2$. For example, an input of 101$\square$1011 will produce an output of 110111. For this exercise, you may also use the Ad machine we constructed in class, that adds two binary numbers. That is, if the tape holds 1101$\square$1011$\underline{0}$ the result will be $\square\square\square\square$100011.

   Solution: The machine will start on the first character of the first number. It moves past the second number and creates two copies of the second number and then moves back to the first number, and subtracts one. Note that the far right number on the tape is the one that will accumulate the final answer. The rest of the work is done in the top middle node, it moves to the third number, does an add, which removes the third number, goes back to the second number and makes another copy of it. Note the shift right to move the sum out of the way of the copied number. The machine then moves back to the first number and subtracts one. This continues until the subtraction produces a space, that is, 0. At this point the machine will remove the next two numbers leaving the last number, which id the product, as the only one on the tape.



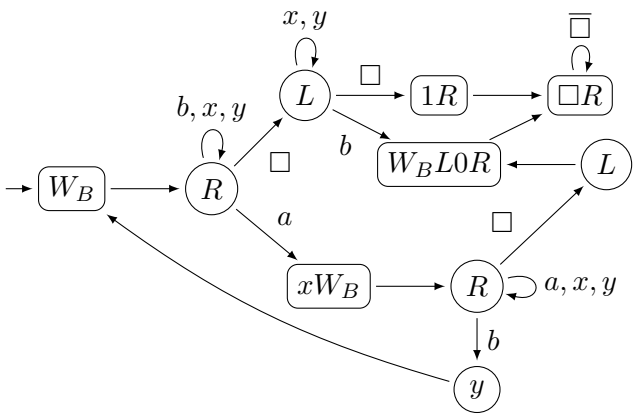
2. Construct a Turing machine, in diagram form, that will take an input of a single word from $\{a, b\}^*$ and write the number of characters in binary on front of the word. The original word is not altered by the computation. For example, if the input tape is $\Box\underline{a}bbbababbaa\Box$ the Turing machine produces $\Box 1011\Box\underline{a}bbbababbaa\Box$. You may use other tape symbols if you would like, in this case the R_x, $R_{\overline{x}}$, L_x and $L_{\overline{x}}$ machines work as usual.

   Solution: The machine starts on the first character of the input word, moves two spaces to the left, writes a 0 and then moves back to the start of the word. If an a is read the machine will write an x, move to the number, add one, and move back to the word. If a b is read the same happens but a y is written instead. The machine will move right until it encounters an a, b, or space. If an a or b, it writes an x or y and the process continues. If it finds a space then the process is over so the machine moves left, replacing x and y by a and b until it hits the space between the word and number, at that point it moves right one space to put the read/write head on the first character of the first word.



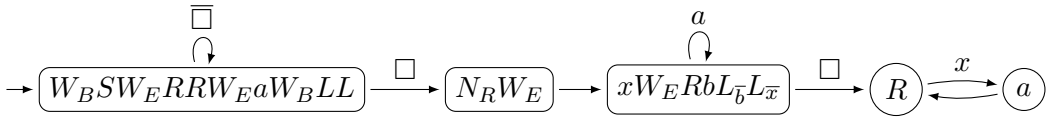
3. Construct a Turing machine, in diagram form, that will leave only a 1 on the tape if the current word on the tape is in the language $L = \{w \mid n_a(w) = n_b(w)\}$ and will leave only a 0 on the tape if the word is not in the language.

   Solution: The machine starts anywhere in the input word and moves the read/write head to the first character. It proceeds to find an a, if it does it marks the a with an x, goes back to the beginning of the word and begins searching for a corresponding b. If it find one, it marks the b with a y, resets the read/write head to the beginning of the string and the process continues. If the machine is searching for a b and encounters a space then there are more a's than b's and the machine goes to the start of the word, writes a 0 and then erases the word. If the machine is searching for an a and encounters a space, it will move back through the word searching for a b. If the machine finds a b then there are more b's than a's and the machine goes to the start of the word, writes a 0 and then erases the word. If the machine encounters a space then there are an equal number of a's and b's, so the machine writes a 1 and erases the rest of the word.



4. Construct a Turing machine, in diagram form, that takes an input number, n, in binary form, and outputs the word $a^n b^n$. For example, an input of 1011 produces *aaaaaaaaaaabbbbbbbbbbb*.

   Solution: The machine starts by subtracting one from the input and then moving to the far right and writing an a. It then moves back to the number and the process continues. Once the number is at 0, that is a space, the machine moves to the string of a's and creates a string of b's of the same size, changing the a's to x's in the process. At the end it converts the x's back to a's and halts.



5. Construct a Turing machine, in diagram form, that takes an input number, n, in binary form, and returns the binary form of 2^n. For example, an input of 101 will produce an output of 100000.

   Solution: The key to this exercise is to notice that, in binary, 2^n is a 1 with n 0's after it. So all that needs to be done is subtract one from the number and add a 0 to the end until subtraction results in a 0, that is, a space. The machine first writes a 1 to the right of the input number, moves back to the beginning of the input number then continually subtracts one and adds a 0 to the end of the result. We assume that the read/write head starts on the first character of the input number.