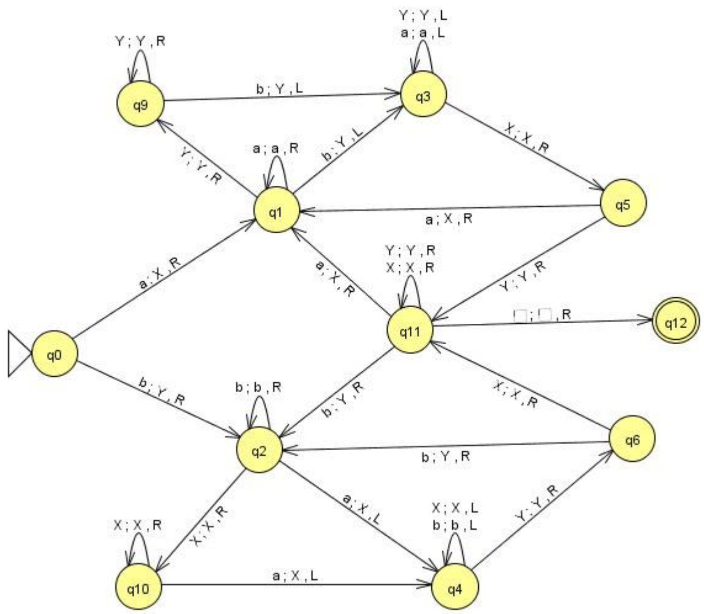
**CS 321 HW 7**

Submit to Canvas a pdf file containing verbal explanations and transition graphs for the Turing machines in problems 1 & 2 and the written answers to problems 3. Also submit JFLAP .jff files (named youronidnameP1a, youronidnameP1b, etc.) for problems 1 & 2.

1. *(10 pts)* Design single-tape Turing machines that accept the following languages using JFLAP

a) L2 = { w : na(w) = nb(w) : w∈$\{a, b\}^+$ }.

| Test case | Result |
|---|---|
| abbaba | accept |
| aaabbb | accept |
| aaaaaabbbbbb | accept |
| ba | accept |
| a | reject |
| abb | reject |
| bbaab | reject |



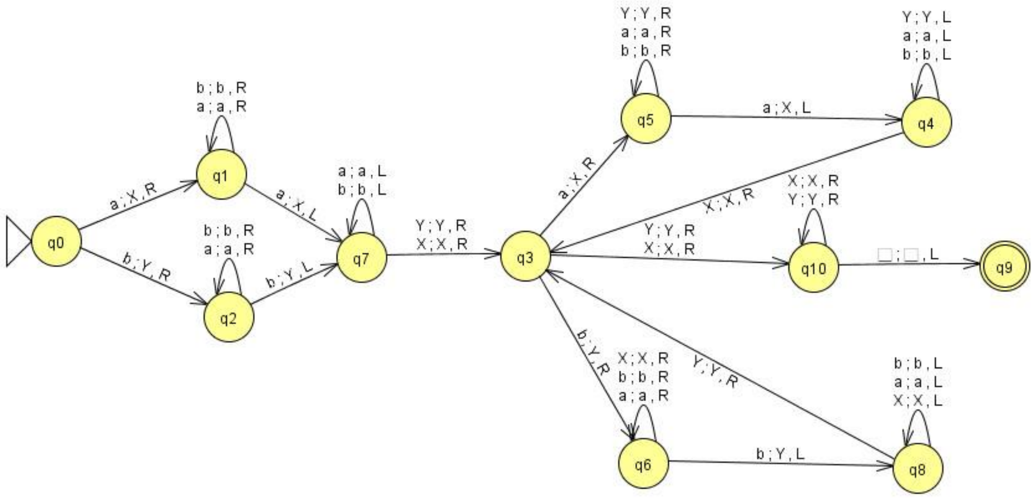

Verbal explanations:

In this question, we know that number of “a” is equal to number of “b” and not allow $\lambda$ in the string.

We start at q0 state, we only allow to read a or b at the first of string it is because it can make sure $\lambda$ is incorrect. On the first case which we read “a” at the string, we rewrite X and go right to state q1. Than we will go right and do not change when we read “a” in the string until we read “b” or “Y”. When we read “b”, we will go left on the string and go to state q3. If we read “Y” in the string, we will keep go right and go to state q9 until we read “b” in the string, we will change “b” to “Y” in the string than go left of string and to go state q3. In the q3, when we read “Y” and “a” in the string, we won’t change and go left in the string until we read “X” and keep “X” in the string, we go to state q5. There is two case in the q5, if we read “a” in the string, we write “X” in the string and go right than go to state q1. If we read “Y”, we will keep go right and to state q11. In the q11, we stay in q11 when we read “X” and “Y” until we read “a” in the string than write “X” in the string and go right and go to state q1. If we read $\lambda$ at state q11, we will go to state q12 and it is final. It is same situation on the case two in state q0.

b) L3 = {ww : w∈{a, b}+ }.

| Test case | Result |
|---|---|
| abaaba | accept |
| bbbbbb | accept |
| aabbaabb | accept |
| a | reject |
| aabb | reject |
| bbb | reject |




My idea is to read first either “a”, or “b” set up the first w, then randomly to find the same “a” or “b” to set up second w. So, it can make sure that first w and second w is same order.

In the beginning (q0, q1, q2, q7), there is to way, which separate “a” and “b”, it is because we need make sure the order of w.

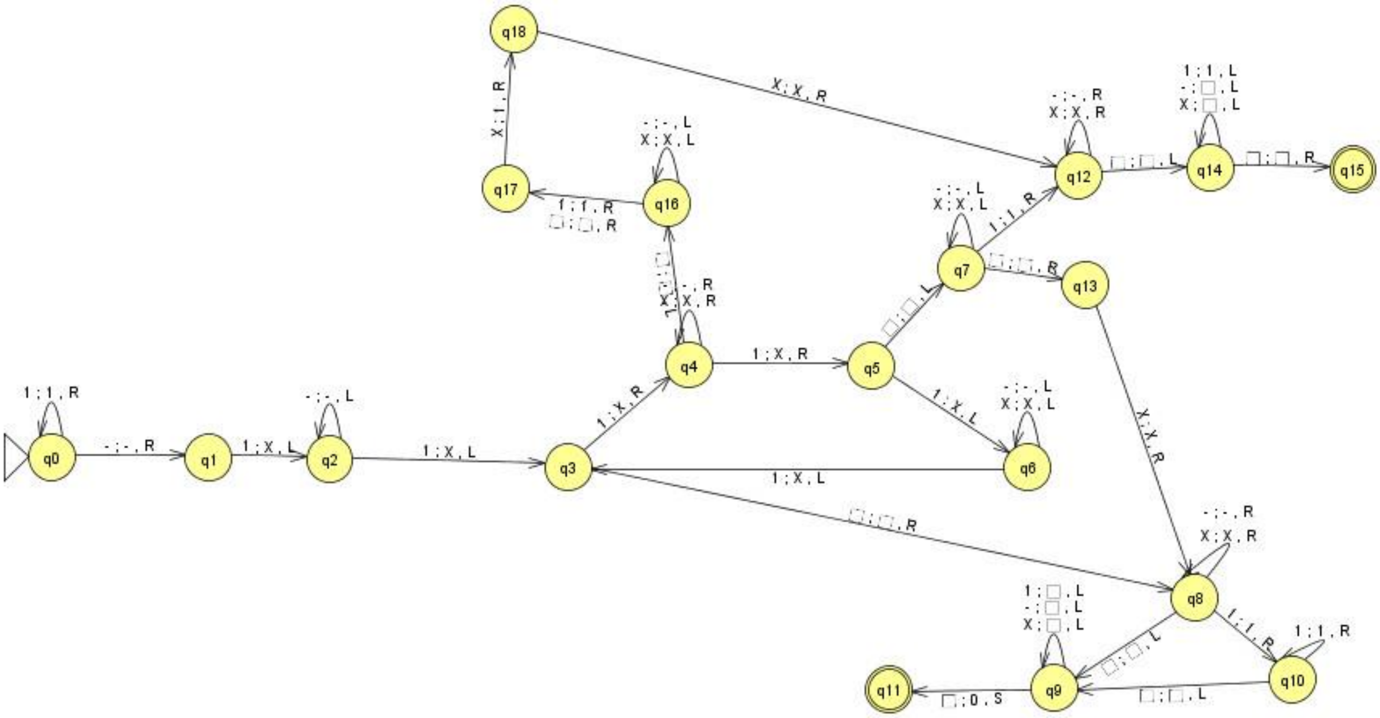
At the second loop (q3, q5, q4, q6, q8), is the same idea but focus on next symbol in the string.

In the end, if we read either “x” or “y” which mean the first w already finish, therefore, we will go to the end of right to check if there still have “a” or “b” lave until we $\lambda$ than go to final state.

2. *(10 pts)* Design Turing Machines using JFLAP to compute the following functions for *x* and *y* positive integers represented in unary. The value *f*(*x*) represented in unary should be on the tape surrounded by blanks after the calculation.

a) $f(x) = \begin{cases} x - y, & x > y \\ 0, & \text{otherwise} \end{cases}$

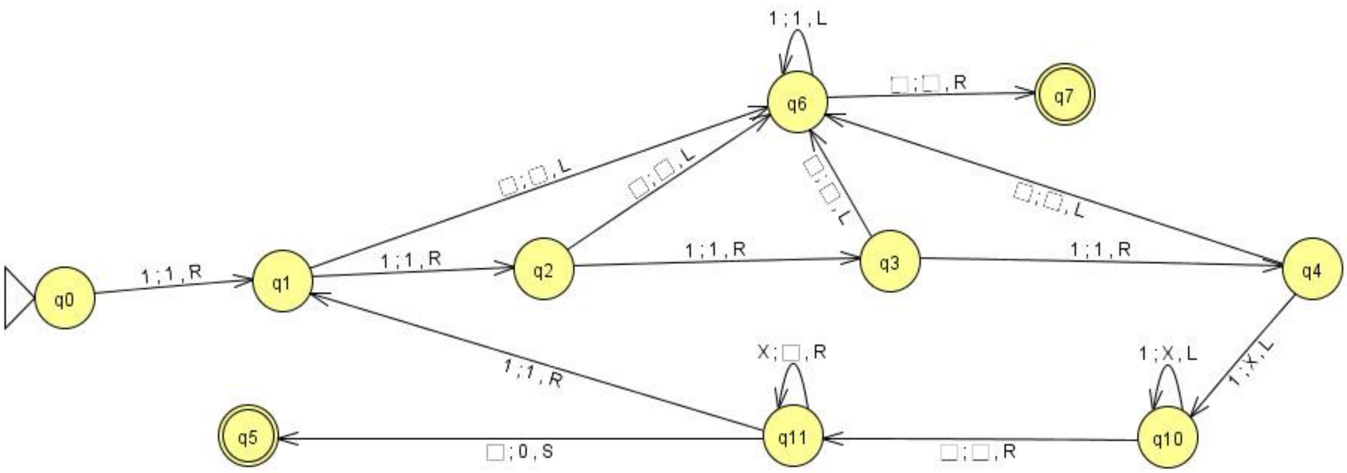
| Input | Output | Result |
|---|---|---|
| 11-1 | 1 | Accept |
| 1-1 | 0 | Accept |
| 111-1 | 11 | Accept |
| 1-1111 | 0 | Accept |
| 1111-11 | 11 | Accept |




My idea in the 2a is that I start change “1” to “X” after “-”. It is because it can check the string is follow the role. Until we find the “1” after “-“, we go back to check if there is “1” in the left of “-” or not. If yes, keep compare the right and left “1” until read the $\lambda$. If write the $\lambda$ at left, go to the end of right and change all “X” and “-” to $\lambda$ than go to the left end. If we read $\lambda$ at the right, I will check the order if there still have “1” at the left, it will same situation before. If not, change everything to $\lambda$ until the right $\lambda$ and put “0”.

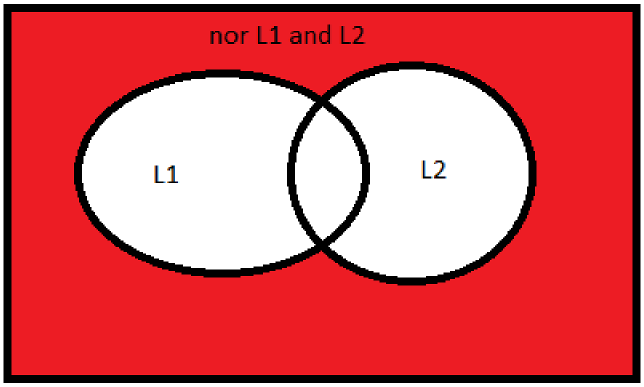
b) $f(x) = x \; mod \; 5$

| Input | Output | Result |
|---|---|---|
| 1 | 1 | Accept |
| 1111 | 1111 | Accept |
| 11111 | 0 | Accept |
| 1111111 | 11 | Accept |
| 1111111111 | 0 | Accept |
| 11111111111 | 1 | Accept |




My idea in question 2b is that when we read $\lambda$ at the end of string, we go back to start and output. If we read the fifth "1", change to "X" than go back to start, rewrite $\lambda$ for first four "1". After that rewrite "X" to $\lambda$, if "1" after that go back to q1, otherwise, rewrite "0".

3. *(5 pts)* The nor of two languages is defined below:
Nor (L1, L2) = { w: w ∉L1 and ∉ L2}.
Prove that recursive languages are closed under the nor operation.




At first, we assume L1 and L2 are recursive languages and they are accept by Turing Machine such as TM1 and TM2.

TM1 and TM2 accept string and halt.