On my honor, I have not given, nor received, nor witnessed any unauthorized assistance on this work.

Print name and sign: _____

Question:	1	2	3	4	Total
Points:	10	13	4	3	30
Score:					

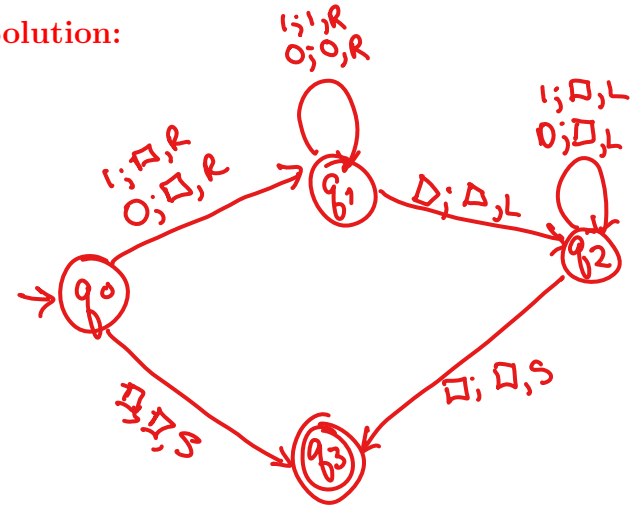
1. (10 points) Draw the machine described by the tuple:

$(\{q0, q1, q2, q3\}, \{0, 1\}, \Gamma, \delta, q0, \{q3\})$

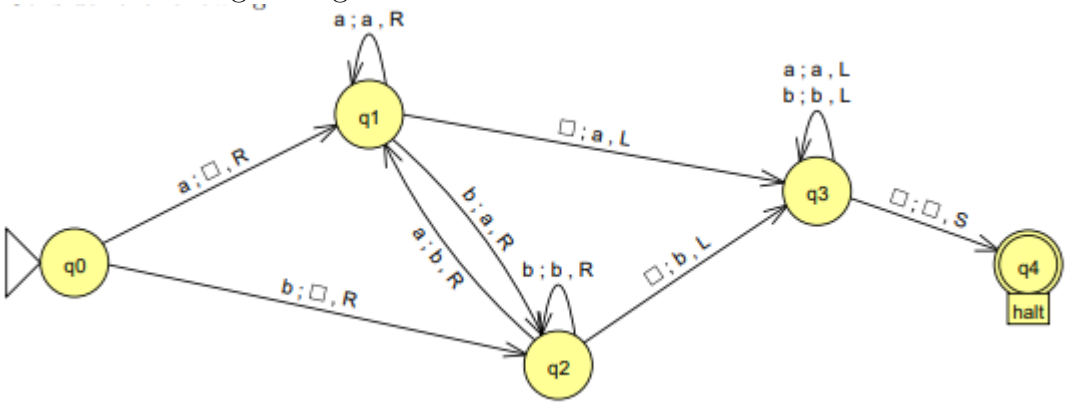
δ:

	0	1	□
q0	(q1, □, R)	(q1, □, R)	(q3, □, S)
q1	(q1, 0, R)	(q1, 1, R)	(q2, □, L)
q2	(q2, □, L)	(q2, □, L)	(q3, □, S)

Solution:



2. Consider the following Turing Machine:



(a) (8 points) The tape contains:

```
... | a | b | b | a | ...
    ^
```

and all other squares are blank and ^ indicating the starting position of the machine's head. Give the final configuration of the machine when it halts. A configuration consists of two things: the state (eg. q0, q1, etc) and the symbols currently on the tape. *Hint, writing down a sequence of configurations is the way to earn partial credit on this problem.*

Solution:

q_0 ..□a bb a□ ..
q_1 ..□□ b b a□ ..
q_2 ..□□ a b a□ ..
q_2 ..□□ a b a□ ..
q_1 ..□□ a b b□ ..
q_3 ..□□ a b b a ..
q_3 ..□□ a b b a ..
q_3 ..□□ a b b a ..

q_3 ..□□a bba ..
q_4 ..□□ a b b a ..

HALT

Notice how series of configs clearly shows original loc. of "abba" on tape and new position.

(b) (5 points) Give a concise English description of what this machine does for any input (not just the one given in the previous part).

Solution: Shifts everything to the right by one space on the tape.

3. (4 points) Design (draw) a Turing Machine which accepts the language defined by the regular expression $ab(a|b)*$. It doesn't matter what is on the tape when the machine halts. *Hint: what is the shortest string we need to accept? Do we care about the rest of the string?*

Solution:

If the string starts w/ ab we halt. We don't have to worry about anything after that because (a|b)* is any combo of a or b...

4. (3 points) The Halting Problem is a classic problem in computer science. Concisely explain what the halting problem is. You do not need to give a formal explanation; a clear explanation in plain English is sufficient.

Solution: pg. 214 in FoC; p173 in IToC. The halting problem is **undecidable** which is a crucial classification of algorithms – ones which we can't compute a yes/no answer for. Simply put, the Halting Problem asks, "If we are given a program and it's input, can we answer yes or no when asked whether or not the program halts when run on the given input?" Keep in mind, this is for all programs, not just a single one. The answer is no, which we can prove via contradiction.

A great explanation with the proof by contraction:
https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101/solving-hard-problems/a/undecidable-problems