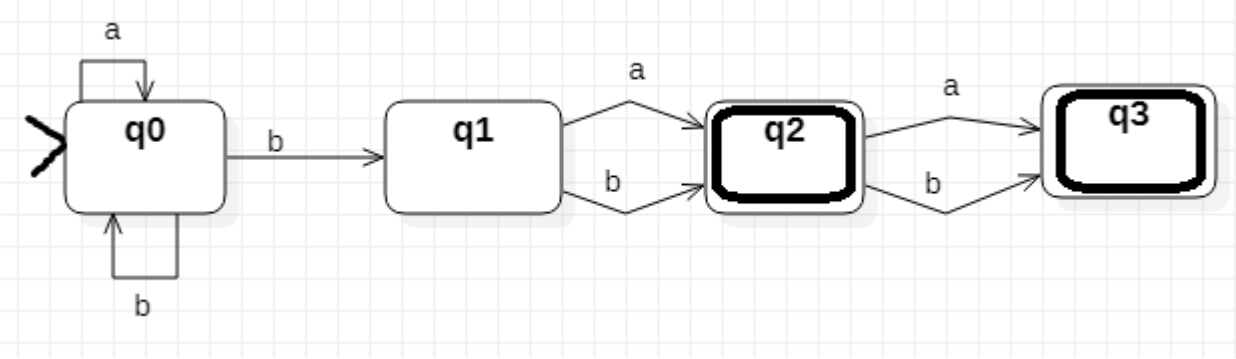
CS332 Mod 4 HW 6

1. (15 pts) Create a nondeterministic finite state machine that accepts strings that contain a *b* in either the second to the last, or third from the last character in the string. For example, strings *ba, ba, baa, bbb, abbbabbaba, abbbabbabaa* would all be accepted. Provide both the graphical form of the machine, and provide M = {Q, Σ, q_0, F, δ}. For reference, Σ = {a, b} and L = *(a + b)*b(a + b) + (a + b)*b(a + b)(a + b)*.

   M = { Q, Σ, q_0, F, δ }
   Q = { q_0, q_1, q_2, q_3 }
   F = { q_2, q_3 }
   δ = (*NOTE: any states referred to as null or not represented in the diagram are implied to represent rejected strings*)

	a	b
q0	0	0, 1
q1	2	2
q2	3	3
q3	Null	Null



CS332 Mod 4 HW 6

2. (15 pts) It is a common student exercise to design a PDA that recognizes palindromes of the form *ucv*, where 'c' is a single element in the alphabet and string *v* is the reverse of string *u*. The purpose of the *c* at the mid-point of the string is to mark the end of *u* so the machine can start processing *v*. This is a convenience. Provide a description of a nondeterministic PDA that can recognize palindromes without the convenient mid-point marker. That is, provide a description of a nondeterministic PDA that recognizes *uv*, where *v* is the reverse of *u*

Assuming that Σ = {a, b}, you probably want to have Γ = { 0, 1, $ } (where Z = $) so that you can use "0" to demark "a" and "1" to demark "b" in the stack.
The main hook of this problem is identifying when to actually start the mirrored side of the palindrome. For a simplified overview of how it may work you can view the diagram below.
I would probably approach this problem by having 2 states:

- q0: which will handle all the inputs for string u.
    - If an "a" is inputted…
        - (No matter what is popped from the stack) anytime an "a" is inputted it can push a "0" onto the stack (along with replacing whatever else was popped) and transition back to q0 where it is ready for another input.
        - OR anytime a "0" is popped from the stack and an "a" is inputted, it can push λ and then transition to q1….
        - If a "1" is popped from the stack, it means there was previously a "b" inputted and thus it certainly is not the beginning of string v and thus it should push back a "1" in addition to a "1" or "0" depending on if an "a" or "b" is inputted. It has to remain in q0 to take another input in this situation .
    - If a "b" is inputted…
        - (No matter what is popped from the stack) anytime a "b" is inputted it can push a "1" onto the stack (along with replacing whatever else was popped) and transition back to q0 where it is ready for another input.
        - OR anytime a "1" is popped from the stack and a "b" is inputted, it can push λ and then transition to q1….
        - If a "0" is popped from the stack, it means there was previously an "a" inputted and thus it certainly is not the beginning of string v and thus it should push back a "0" in addition to a "1" or "0" depending on if an "a" or "b" is inputted. It has to remain in q0 to take another input in this situation.
- q1: which will handle all the inputs for string v.
    - In this state it has limited options for what it can input/pop. It must input based on what it pops ("a" for "0" and "b" for "1"). It has to continue this until it runs out of stuff to pop , where then it means that the string u has been fully mirrored as string v.