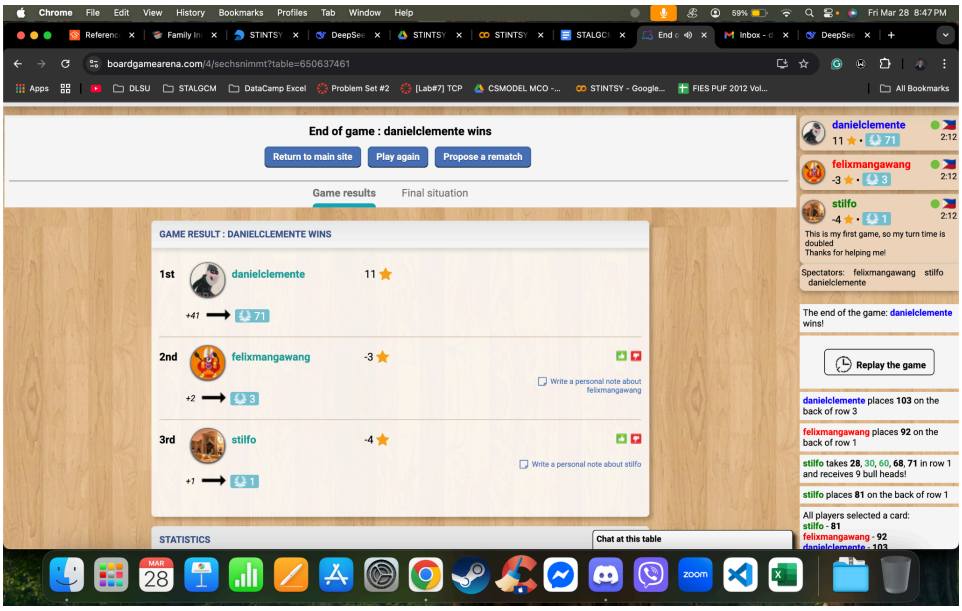
Members	Section	Group Number	6nimmt! ID
Daniel Clemente	S17	25	danielclemente
Joshua Laxa	S17	25	stilfo
Felix Mangawang	S15	25	felixmangawang

1. (20 points + 1 point bonus) Model the turn-based mechanics and key events of the board game 6nimmt! using a PDA. This will involve identifying states, transitions and symbols based on the rules of the game. Play at least one full game of the online version (https://en.boardgamearena.com/gamepanel?game=sechsnimmt) with the default setup. For solo players, you are allowed to play with other individuals (indicate whether they are relatives, non-DLSU friends, DLSU students, or DLSU personnel).
    a. (1 point bonus) Provide a screenshot of the final game result.



    b. (4 points) From which perspective—one player, all players, or a spectator—can we represent a full game using a 1-stack PDA? Choose at least one and explain your answer. Assume that a game has at least three players.

We can represent a full game using a 1-stack PDA from one player's perspective. A 1-stack PDA can suffice for one player's perspective because the player's decisions depend only on their partial observation of the game, which can be managed with a stack.

    c. (8 points) If a full game will be expressed using a 1-stack PDA, what are your input symbols and stack symbols? Explain your answer and provide sample transitions (showing scan, push and pop).

i. Example: The input symbols represent the player IDs while the stack symbols represent the player ratings because <explanation>.

To model 6nimmt! with a 1-stack PDA, the input symbols represent player actions (e.g., c_x for playing card x, r_i for selecting row i, t for resolving turns, and # for game end), while the stack symbols track game state (e.g., (row_i, x) for card x in row i, | as a row separator, and $ for stack bottom). For example, when a player plays card 25 (c_25), the PDA pushes (row_2, 25) onto the stack; when resolving a turn (t), it pops cards until a separator to check for row limits, enforcing the game's rules. This approach leverages the stack's LIFO structure to simulate row management, though a queue would better match the game's FIFO eviction rule.

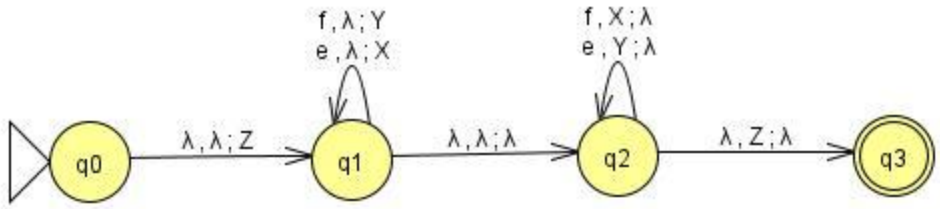
d. (4 points) If another stack will be introduced, how will you use the first stack and how will you use the second stack?
   i. Example: The first stack tracks the player ratings while the second stack tracks the player's IDs.

   With two stacks, the first stack manages the live game state (tracking rows and card placements), while the second stack logs the sequence of player moves for validation or replay. This split keeps the automaton efficient—one stack handles real-time rules while the other ensures accountability. Two stacks balance gameplay simulation with record-keeping, avoiding the limitations of a single stack.

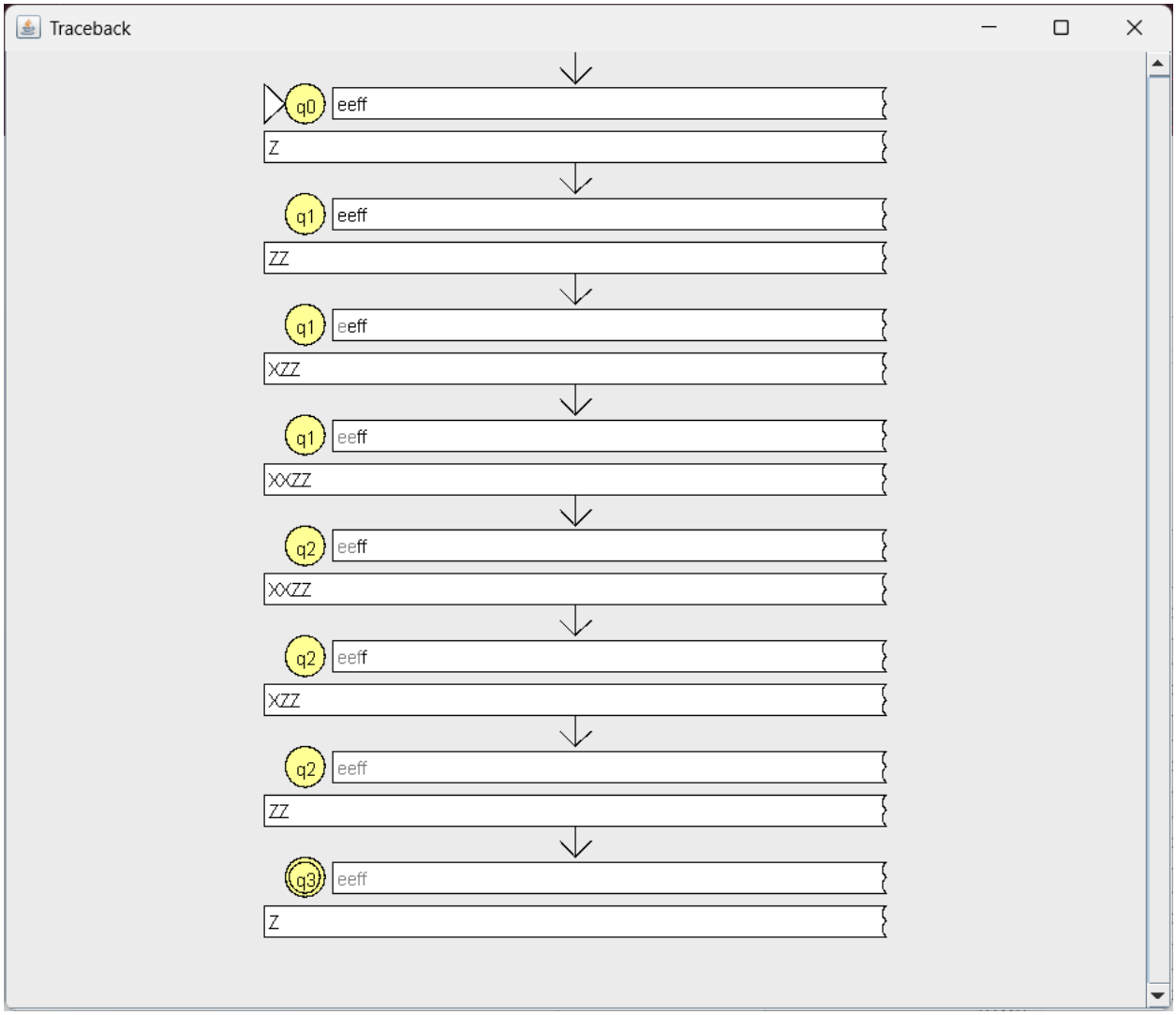
e. (4 points) If you have the power to change the data structure used by the PDA, which data structure will you use and why?

   The data structure I would use is a Queue. A queue is the clearest choice for modeling *6nimmt!* because it perfectly matches how the game works—when a row gets too long, the oldest card (first in) is the one that forces a player to take the pile. This keeps the automaton simple and deterministic, without unnecessary complexity. Unlike a multiset, which ignores order, a queue preserves the exact sequence of plays, which is crucial for resolving turns correctly.

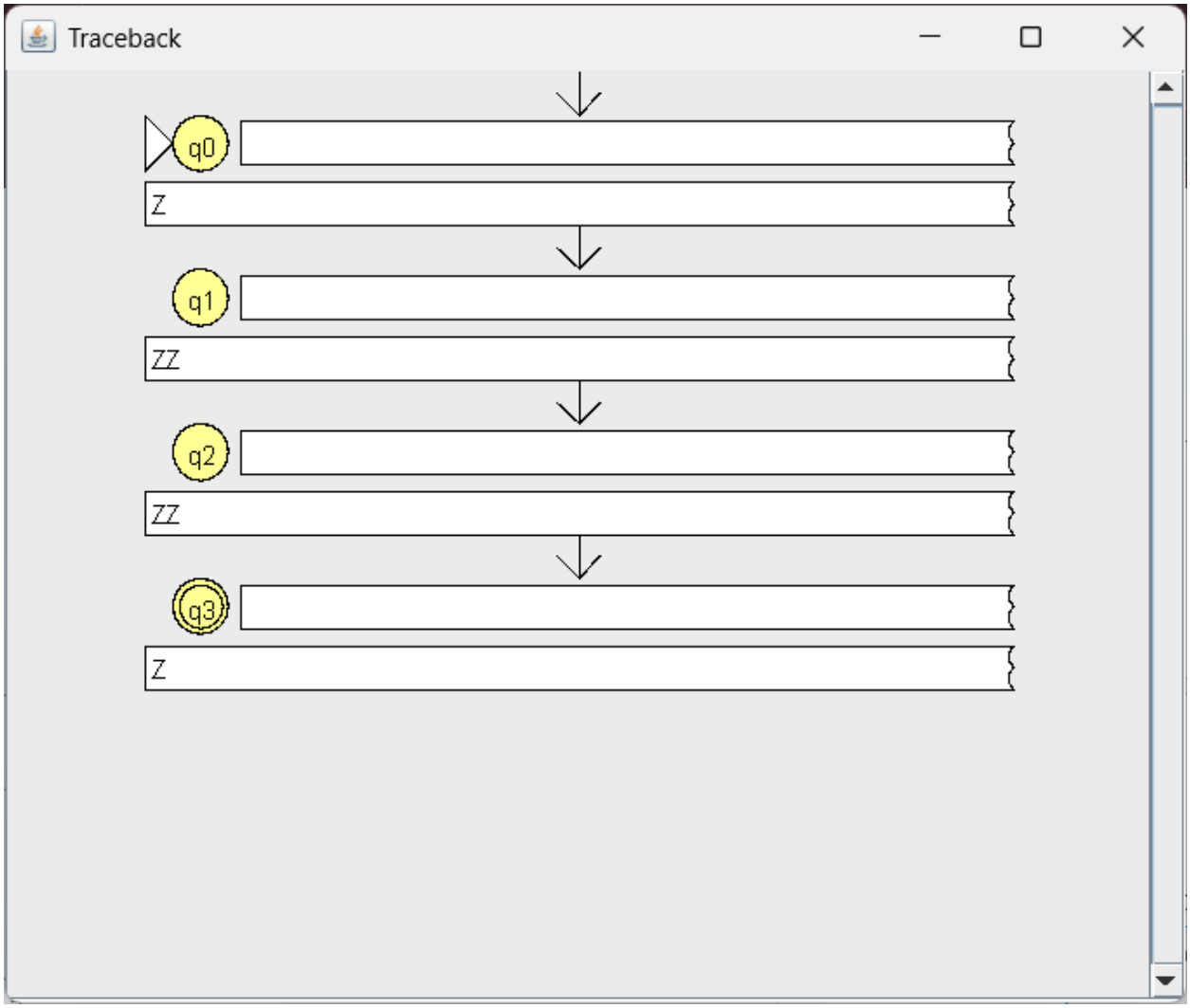
2. (15 points) Using the convention discussed in class (operations Scan, Push, Pop), design a PDA that accepts strings where the number of English words (e) is equal to the number of Filipino words (f). The input can be an empty string.


f , λ ; Y
e , λ ; X
f , X ; λ
e , Y ; λ
λ , λ ; Z
λ , λ ; λ
λ , Z ; λ
q0
q1
q2
q3

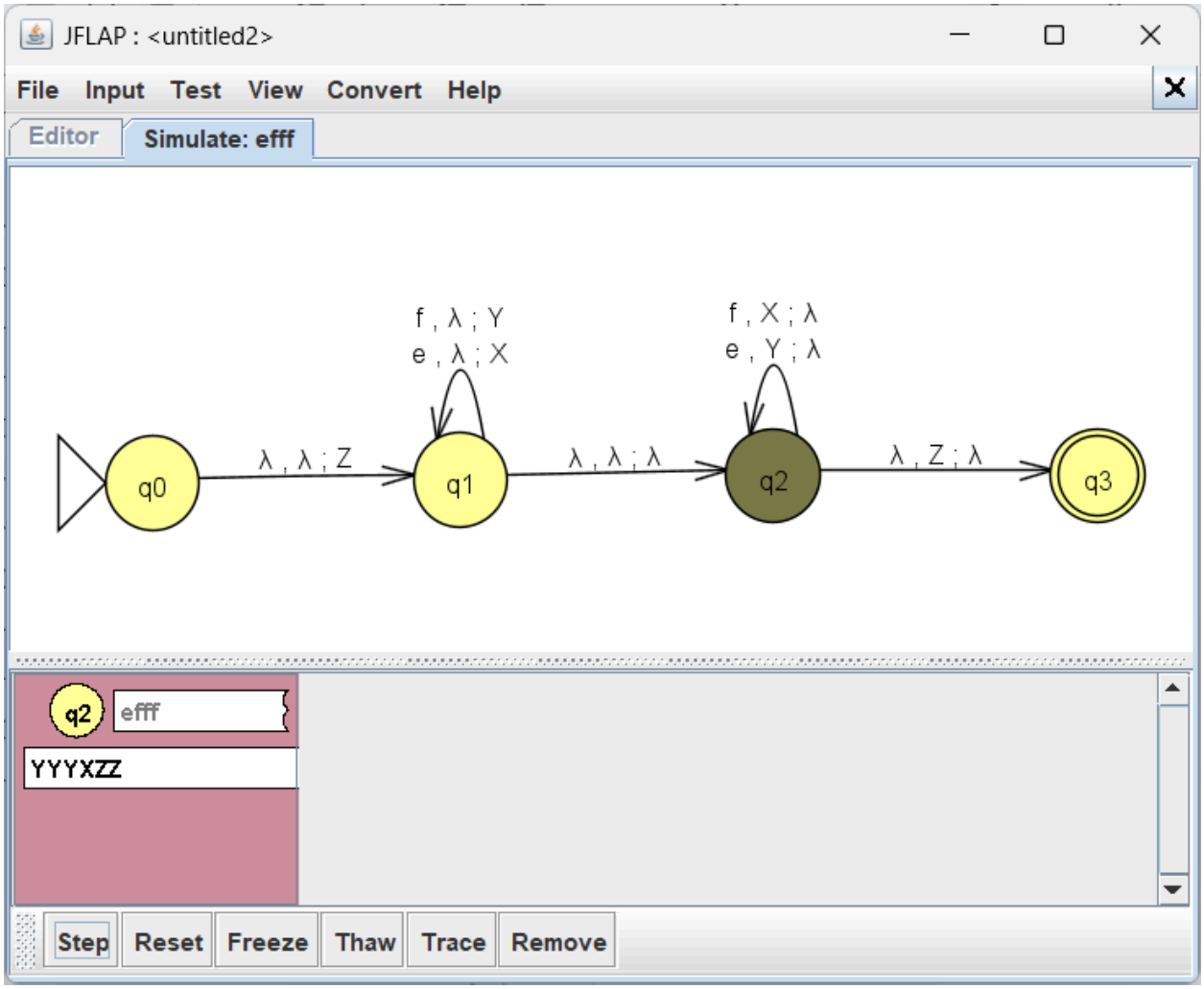
Accepted: eeff



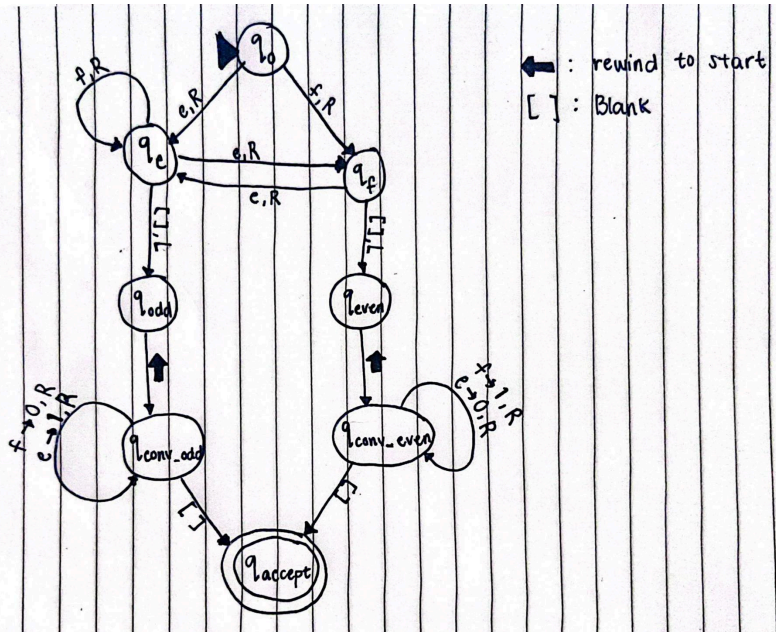
Accepted: empty input string



Rejected: eefff



3. (15 points) Design a Turing machine that converts all English inputs (e) to 0 and all Filipino inputs (f) to 1 if the number of English inputs (e) is even. If it is odd, convert all English inputs (e) to 1 and all Filipino inputs (f) to 0. Assume that the input contains at least one character.


: rewind to start
[ ] : Blank
q_0
q_e
q_f
q_{odd}
q_{even}
q_{conv_odd}
q_{conv_even}
q_{accept}
f,R
e,R
f,R
e,R
e,R
[],L
[],L
f→0,R
e→1,R
f→1,R
e→0,R
[],L
[],L