DUE WEDNESDAY 21 MARCH

OH GOD, WHYYYYY?!?!?!?!

- File input and main vector initializations (Main)
    - Run executable (with an optional file argument)
    - Parse argument
        - If no argument, prompt for CSV file
    - Check if CSV file
        - Re-prompt if invalid
    - First line has names of candidates, initialize vector of Candidate classes with names of candidates
    - Prompt for number of seats to fill
        - If seats are more than the number of candidates, raise an error
    - Run through file from 2nd line and add each line to vector of Ballot classes (names and corresponding rankings, and number of ranked votes)
    - Prompt 1 for plurality and 2 for droop

- Plurality
    - Iterate through Ballot vector
    - Pop off current Ballot and add it to the corresponding Candidate's Ballot using addBallot() (highest ranking). This removes it from the main Ballot list and stores it with its candidate
    - After all ballots have been iterated through, check the candidates and see who has the most
    - Keep doing this until all seats are filled
    - If there's a tie, choose randomly

- STV
    - Since we have the number of ballots (Ballots.length), calculate droop quota
    - Shuffle Ballots vector

- Call STV method, passing in the main Ballots vector as an argument
- Iterate through main Ballot vector
- Pop off current Ballot and add it to the corresponding Candidate’s Ballot using candidate.addBallot() (highest ranking). This removes it from the main Ballot list and stores it with its candidate
    - If highest-ranked candidate is not on the main Candidate vector, check for the next-highest ranked
    - If none of the preferred candidates are on the main vector, discard current ballot (delete) and move on
- If candidate reaches droop, pop off candidate and add it to Winners vector
- After all Ballots have been iterated through, check to see who has lowest vote count (Candidate.Ballots.length)
- Pop off loser and add to Losers vector
    - If tied, randomly pick
- Recursively call STV method with loser’s Ballot vectorm
- Keep doing this until main Candidate’s vector is empty (all are assigned as winners or losers)
- Iterate through winners until seats are filled
    - If tied, pick who won first
- If there aren’t enough winners to fill all seats, iterate through losers BACKWARDS
    - If tied, randomly pick

- Classes
    - Main
        - Ballot vector
        - Candidate vector
        - Winners vector (of type Candidate)
        - Losers vector (of type Candidate)
        - Reads and parses file to fill up Ballot and Candidate vectors

    - Ballot
        - Map of candidate names and corresponding ranking
        - ID number (index in Ballots vector, AKA order of appearance)

    - Candidate
        - Name
        - Ballot vector
        - Void addBallot(Ballot vector) - Associates a ballot with the candidate. Takes in a ballot type and appends it to Ballot vector.
        - Int getVotes() - Returns number of votes. Simply return Ballot.length

    - Algorithm (interface)
        - Acts as an interface for both voting algorithms. Thus, it is a PURE VIRTUAL (ABSTRACT) class.
        - Void vote() - Method used to parse ballot information (DECLARATION)
        - Void findWinner() - Finds winner.

    - Plurality
        - Void vote(Ballot vector, Candidate vector) - Iterates through the Ballots and associates them with their corresponding candidates (whoever is ranked the highest)
        - {Candidate vector} findWinner(int seats, Candidate vector) - Iterates through candidate vector and finds who has the most votes. Keeps doing this until all seats are filled. Returns a vector of winners.

- STV
    - Void vote(Ballot vector, Candidate vector, Winners vector, Losers vector) - Calculates droop quota and iterates through Ballot vector, assigning each to corresponding candidate (see Code Flow section above). Automatically removes Ballots and Candidates, assigning Candidates to Winners if droop is reached, and the lowest votes to Losers. Recursively called using the Loser's corresponding Ballots passed in as the first argument. Stops recursion once Candidate vector is empty (meaning everyone has been assigned as either a winner or loser).
    - {Candidate vector} findWinner(int seats, Winners vector, Losers vector) - Finds the winners for the number of seats. First checks the Winners vector in order, then the Losers vector (BACKWARDS) if needed. Returns a vector of Winners.