Bug List

Names: Jashwin Acharya (achar061), Carlos Chasi-Mejia(chasi009), Vincent Hoang(hoang317), Steve Petzold(petzo017)
Team: 08

Bug #	Description of the bug	Location of the bug	Steps to recreate the Bug and / or Test Case #	Root Cause Analysis Notes
1	For Instant Runoff Voting, currently, if we have two candidates tied for the lowest votes during redistribution, we aren't sure if we need to do a tie-breaker for eliminating one of those candidates. For now we just eliminate the candidate who first occurs in the candidates list who has the lowest number of votes. For example: if our candidates list is [Rosen, Royce, Kleinberg, Chou] and Kleinberg and Chou are tied with the lowest votes during redistribution, our algorithm will eliminate Kleinberg first since his name comes before Chou in our list of candidates and we don't do any tie breaker to judge who should be eliminated. Currently, we only do tie-breakers when we have to find a	The bug is located in the findCandidateForElimination() function located in the IRVoting.java class	You can copy the following lines into a CSV file and run the program and you will see in the audit file that Chou is always eliminated first and a tie-breaker is not performed since Chou and Kleinberg are tied for the lowest number of votes: **IR** **4** **Rosen (D), Royce (L), Chou (I), Kleinberg (R)** **11** **1,3,4,2** **1,,2,** **1,2,3,** **3,2,1,4** **,1,2,3** **,1,2,** **,1,,** **„1,2** **„„1** **„2,1** You can run the code using the following commands: javac VoteEasy.java java VoteEasy And then you can take a look at the generated	Our idea is to slightly modify the code for findCandidateForElimination() by collecting Candidate objects who are tied for the lowest number of votes in an arraylist and then using the SecureRandom class to randomly select a Candidate from the arraylist to eliminate from the election.

	winner for the election. We aren't exactly sure if this is a bug but wanted to record it regardless.		audit file (audit_file.txt) to see that Chou is eliminated first. If you run the program multiple times, you can see that Chou is always eliminated first since they come before Kleinberg in the list of candidates we store in IRVoting.java.	
2	Currently, we keep the party names abbreviated as "D", "R" etc when displaying the result of the election on the UI as well as when writing election results to the audit file. Again we are not sure if this is a bug and if we need to spell out the Party names such as Democrats, Republicans, etc. The reason we don't spell them out correctly is because we aren't sure how many different party names are possible since we don't have a predefined list of party names available to us. Hence we only use initials.	The root of the bug is in the Candidate and Party classes where we don't map Party initials such as "D", "R" etc to their fully spelled names when initializing the party names in the class constructors.	You can copy the following lines into a CSV file and run the program with this file: **IR** **4** **Rosen (D), Royce (L), Chou (I), Kleinberg (R)** **11** **1,3,4,2** **1,,2,** **1,2,3,** **3,2,1,4** **,1,2,3** **,1,2,** **,1,,** **,,1,2** **,,,1** **,,2,1** You can run the code using the following commands: javac VoteEasy.java java VoteEasy You will notice in the results displayed on screen that the party names are not completely spelled out and you can see the same in the generated	We can slightly modify the code in the constructors for classes Candidate and Party to map party initials to their actual spelled-out names. The Java Map class can easily be used to resolve this issue.

			audit file (audit_file.txt) too.	
3	For Open Party List, there is a bug for the case where the number of ballots is less than the number of seats which causes there to be more remaining seats left to redistribute than there are parties left after the first round of seat allocation. We don't know for certain if this is a case to consider, but when this happens, then a runtime error occurs with the message "Index 0 out of bounds for length 0."	The root of the bug is in the determineRemainingSeatAllocation() method in OPLVoting.java class.	You can copy the following lines into a CSV file and run the program: **OPL** **6** **Pike (D), Foster (D), Deutsch (R), Borg (R), Jones (R), Smith (I)** **13** **9** **1,,,,,** **1,,,,,** **,1,,,,** **,,,,1,** **,,,,,1** **,,,1,,** **,,,1,,** **1,,,,,** **,1,,,,** You can run the code using the following commands: javac VoteEasy.java java VoteEasy	We can modify the code by having it redistribute the rest of the remaining seats again to all the parties that have already received a remaining seat so we don't run into this out of bounds error.
4	For Open Party List, there is a potential bug where a party receives more seats than it has candidates. We don't know for certain if we would have to do anything special if this case occurs, but for now we still give the seats to the party even if they have more seats than candidates.	The root of the bug is in the performSeatAllocations() method in OPLVoting.java class.	You can copy the lines in OPL_test_large_number_of_votes.csv into the /src directory and run the program using the following commands: javac VoteEasy.java java VoteEasy And then you can take a look at the generated audit file (audit_file.txt) to see that the I party won 2 seats, but it only has one candidate.	We can modify our code to check the number of candidates that a party has before giving it a seat if it exceeds the number of candidates that are in that party.