|  |  |
| --- | --- |
| **Project Name: Aegis Project 1: Voting System Team#7** | |
| **Test Stage: Unit X System \_** | **Test Date:** 3/23/2021 |
| **Test Case ID#:** PartyTest\_7: sortByVotesPreReverseOrder1 | **Name(s) of Testers:** Michael,Donald,Grant,Lucky |
| **Test Description:**  Verifies that sortByVotes() is sorting the candidates by their number of ballots from least to most. The vector being sorted is already in reverse sorted order. |  |
| **Automated: yes X no \_\_\_** | **Indicate where you are storing the tests (what file) and the name of the method/functions being used.**  Stored in file: PartyTest.cpp  Method names:   * sortByVotes() * addCandidate() * addBallot() (used to create Derrick3) * getPartyMembers() |
| **Results: Pass \_\_X\_\_\_** Fail **\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**  Party \*D, Candidate \*Derrick, and Candidate \*Derrick2 have all been setup properly. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
|  |  |  |  |  |  |
| 1 | Properly initialize a third candidate and add to party. | Party D  Party D2  D2->getPartyMembers()  Candidate \*Derrick3 |  |  |  |
| 2 | Create another party and manually add in candidates such that they’re all in reverse sorted order. |  |  |  |  |
| 3 | Call D2->sortByVotes() |  |  |  |  |
| 4 | Iterate through each candidate in the partyVector and verify that the candidate’s ballots are less than or equal to its own. If one comparison fails, the whole test fails. |  | D->getPartyMembers() = {Derrick,Derrick2, Derrcik3} | D->getPartyMembers() = {Derrick,Derrick2, Derrcik3} | The ballots are all sorted correctly. |

**Post condition(s) for Test:** 

There have been no changes made. sortByVotes() has been run on a Party D2.