|  |  |
| --- | --- |
| **Project Name: Project 1: CompuVote Team #19** | |
| **Test Stage:** Unit | **Test Date:** 03/14/2021 |
| **Test Case ID #:** Test\_393\_07\_01 | **Name(s) of Testers:** Jack Fornaro |
| **Test Description:**  Given a standard IR election with candidates, parties, and ballots, where there is one clear candidate with the highest votes and one clear candidate with the lowest votes, tests that getLowestHighestCandidates returns one lowest and one highest candidates |  |
| **Automated:** Yes | **Indicate where you are storing the tests (what file) and the name of the method/functions being used:**  Test file: Project1/src/test/org/team19/InstantRunoffSystem.java  Test method: testGetLowestHighestCandidatesSingleHighestSingleLowest  Method/constructor being tested: getLowestHighestCandidates from Project1/src/main/org/team19/InstantRunoffSystem.java |
| **Results:** Pass |  |
|  |  |
| **Preconditions for Test:**  N/A | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
|  |  |  |  |  |  |
| 1 | Tests a pair containing Rosen (highest) and Kleinberg (lowest) are returned by getLowestHighestCandidates | Candidates:  Rosen (D) – 3 ballots  Kleinberg (R) – 0 ballots  Chou (I) – 2 ballots  Royce (L) – 1 ballot | “Pair{Pair{0, [Kleinberg (R)]}, Pair{3, Rosen (D)}}” | “Pair{Pair{0, [Kleinberg (R)]}, Pair{3, Rosen (D)}}” | N/A |

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

N/A