| **The PBI** : 13  **Task Description (from Sprint Log):** Below is the log for Tie Breaker between Parties IR  **Unique ID** : TieBreakerPartiesIR |
| --- |
| Team Member(s) Responsible: Luke Chen |
| Inputs:  testing/irTest1.csv,  testing/irTestFile1.csv,  testing/irtest.csv,  testing/irTie.csv |
| Tests:  testFilter1, testFilter2, testFilter3, testFilter4 |
| Outputs:  testFilter1 pass: 5  testFilter2 pass: 19  testFilter3 fail: -1 due to empty csv  testFilter4 fail: -1 |
| Passed or Failed:  2 pass  2 fail |
| Date:  4-29-21 |

| **The PBI** : 1  **Task Description (from Sprint Log):** Below is the taste log for Tie Breaker between Candidates  **Unique ID** : Tie\_Candidates |
| --- |
| **Team Member(s) Responsible**: Abdikarim Fareh |
| **Inputs**: testing/opl3.csv  I used TieBreakerOPLCandidatesTest() Java Class for the JUnit test |
| **Tests**:   * When candidates of the same party have same total vote count, we break the tie between them * If two candidates have same number of votes but there is 2 seats, we will automatically pic them |
| **Outputs**:  1st run - winners are : Joe, Luke, Billy, Isaac  2nd run - winners are : Joe, Luke, Billy, Sandra   * In the case above, the party “Independent” has 1 seat only but there are two candidates that have the same result, so we break the tie. |
| Passed or Failed:  **Pass** |
| Date: **4/27/2021** |

| **The PBI** : 1  **Task Description (from Sprint Log)** : Below is the log for Tie Breaker between parties.  **Unique ID** : Tie\_parties |
| --- |
| Team Member(s) Responsible: **Abdikarim Fareh** |
| **Inputs**: testing/opl2.csv  I used TieBreakerOPLPartiesTest() Java Class for the JUnit test |
| **Tests**:   * When parties do not have enough candidates they deserve to get a seat. Since they do not have a candidates to occupy the seats, we distributed to next party * An actual tie between parties |
| **Outputs**:   * 1st run   + Party R has 2 seats   + Party D has 0 seats   + Party I has 2 seats * 2nd run   + Party R has 2 seats   + Party D has 1 seats   + Party I has 1 seats * The above output is when we have a tie between “D” and “I” in which they both have the same number of votes. So, we break the tie. At the same time we are making sure that they have enough candidates for their allocated seats |
| Passed or Failed:  **Pass** |
| Date: **4/27/2021** |

| The PBI :5  Task Description (from Sprint Log): PBI 5 Code OPL for multiple OPL files  Unique ID : Multiple\_OPL |
| --- |
| Team Member(s) Responsible: Rohan Abraham |
| Inputs: file1.csv |
| Tests: Tests 1 and 0 |
| Outputs: “100” and 10 |
| Passed or Failed: Pass |
| Date: 4/28/2021 |

| The PBI : 5  Task Description (from Sprint Log) : PBI 5 Code IR tests multiple files for IR  Unique ID : Multiple\_OPL |
| --- |
| Team Member(s) Responsible: Rohan Abraham |
| Inputs: file6.csv |
| Tests: test0 and test1 |
| Outputs: “40” and 5 |
| Passed or Failed: Pass |
| Date: 4/28/2021 |

| The PBI : 5  Task Description (from Sprint Log) : PBI 5 Code PO test multiple files for PO  Unique ID : Multiple\_PO |
| --- |
| Team Member(s) Responsible: Rohan Abraham |
| Inputs: file5.csv |
| Tests: test0 and test1 |
| Outputs: ”40” and 5 |
| Passed or Failed: Pass |
| Date: 4/28/2021 |

| The PBI : PBI 5 Code Main  Task Description (from Sprint Log) : PBI 5 Code Main tests code for multiple files works in main  Unique ID : Multiple\_Main |
| --- |
| Team Member(s) Responsible: Rohan Abraham |
| Inputs: file1.csv, file2.csv |
| Tests: test3 and test4 |
| Outputs: “110”, “end file2”, “END FILE 3” |
| Passed or Failed: Pass |
| Date: 4/28/2021 |

| The PBI :  Task Description (from Sprint Log) :  Unique ID :  PBI 3. Implement Code for PO to read in the data from the csv file. |
| --- |
| Team Member(s) Responsible:  Isaac Xiong |
| Inputs: poTie.csv, po.csv, poZero.csv |
| Tests:  Run votecountsystem with a popularity only csv file. The program will run and read the data from the PO csv file. |
| Outputs:  poTie.csv: returns the audit and results of the PO election. There is a random tie breaker between two candidates and one candidate is chosen as the winner. This is recorded in the audit and results file. All the data is read.  po.csv: returns the audit and results of the PO election. The winner of this election is “Joe”, This is not recorded in the results file, but is in the audit file. This is a bug, because there is a winner, but it is not directly being recorded in the results file that is created. All the data is read.  poZero.csv: Reads all the data and prints out the candidate with their number of votes, which is all zero because this file has 0 votes. All the data is read, but program segfaults when trying to compute a winner. |
| Passed or Failed:  *A pass is defined as the program successfully reading in the data from the csv file.*  poTie.csv: PASS  po.csv: PASS  poZero.csv: PASS |
| Date: 4/28/2021 |

| **System Testing** : OPL  **Description** : Testing OPL’s functionality  **Unique ID** : OPL\_Test |
| --- |
| **Team Member(s) Responsible:** Abdikarim |
| **Inputs**: testing/opl.csv |
| **Tests**: Testing to run opl file and be able to announce winners :   * Candidates : 5 * Seats : 3 * Ballots : 100,000 * Party Seats :   + R : 1   + D : 1   + I : 1 |
| **Outputs**:   * Winner : Luke (R), Billy (D), Sandra (I) |
| Passed or Failed: **Pass** |
| **Date**: April 28 2021 |

| System Testing : IR test of the class  Description : Normal IR Run  Unique ID : IR\_Test |
| --- |
| Team Member(s) Responsible: Abdikarim |
| Inputs: “testing/irtest1.csv” |
| Tests: Using 5 candidates and 10000 ballots |
| Outputs:  Billy,20000, Winner  Sandra,20053, Winner  Luke,20061, Winner |
| Passed or Failed: Pass |
| Date: 4/28/2021 |