|  |  |
| --- | --- |
| **Project Name: Project 2: Voting System Team# 25** | |
| **Test Stage: Unit \_1\_ System \_\_** | **Test Date: 12/13/2019** |
| **Test Case ID#: OPL\_001** | **Name(s) of Testers: YongFeng Ji** |
| **Test Description: To test all the function in CPL** |  |
| **Automated: yes\_1\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.** |
| **Results: Pass \_\_\_1\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**  **The input files exist (cplscenario1.csv), and pre-processing has already been done on the file** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
|  |  |  |  |  |  |
| 1 | CPL\_Increment test, to test if the increment function can successfully assign votes to each party | The party vector, provided by the function “get\_parties” | The corresponding candidate’s vote should be increased by one. If the number is over the index, return -1  For example: If parameter is 1. The second party’s voting count in the parties vector should be increased by 1 | The corresponding candidate’s vote should be increased by one.  If the number is over the index, return -1  For example: If parameter is 1. The second party’s voting count in the paties vector should be increased by 1 | Looks good |
| 2 | CPL\_Parties\_test, test if the get\_Parties function works or not, simply compare the result with given name. | The CSV file. | Vector contains all parties, each party include corresponding candidates. the party in the vector should be same as the CSV file  Parties1 contains 6 parties, the fourth party’s second candidates should be “b”. | Vector contains all parties, each party include corresponding candidates. the party in the vector should be same as the CSV file  Parties1 contains 6 parties, the fourth party’s second candidates is “b”. | Looks good |
| 3 | CPL\_GenerateRandomNumber\_Test is to test the function can generate random number or not. Manual check | Test with seed parameter | A random number | A random number | Looks good |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | to test Calculate\_winner works properly or not, we manual check it. | Seed parameter. | The party’s voting count, which party get the most votes should be returned. | The party’s voting count, which party get the most votes is returned. | Looks good |

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

All function works fine, should be ready to go with processor.

Project Name: The project #, name of your system, and the team#

Test Stage: Indicate whether it is a unit test or a system test.

Test Date: The date the test was performed.

Test Case ID#: A unique ID is required. Decide on a naming convention and use numbering. Example: Ballot\_Shuffle\_1

Name(s) of Testers: List the names of anyone involved in running this test case.

Test Description: Describe briefly the test objective.

Automated: Indicate if the test is completely automated or being checked manually. (If you have methods running the tests and checking results, select “yes”. If you are manually checking results, indicate manual by selecting the “no.”)

**Results:** Indicate if the test passed or failed.

**Step #:** You will be listing the test steps in order. This number is the step number in the process.

**Test Step Description:** Details of the test step.

**Test Data:** What the test data will be for this step. Be clear on what the input data will be. If using a specific file, be clear on the name.

**Expected Result:** What result are you expecting from the program component or system.

**Actual Result:** What result were returned based on the test.

**Post condition for Test:** What will be true after the test has been run? Has the state of the system changed in any way?

**Notes:** Comments and notesfor you and your team members.