Project Name: Project 1: Voting System **Team 21**

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 1 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to test a setter and getter function for the parties' names. All the used methods are in the test file named "party_utest_0.cc" and are listed as follow:

1. SetPartiesNames
2. GetPartyName

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with an empty vector of strings containing the names should also be initialized with a size that is equal to the number of parties.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call SetPartiesNames	A vector of strings of parties names {"Democratic", "Republican", "New Wave", "Reform", "Green", "Independent"}	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
2	Call GetPartyName six times and pass in indexes of the parties		Democratic Republican New Wave Reform Green Independent	Democratic Republican New Wave Reform Green Independent	

Postcondition(s) for Test: Names of the parties are stored in a vector of strings in the correct order and can be accessed anytime without having to worry about getting an out of range error of the vector.

—---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 2 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to test a setter and getter function in the class for the candidates' names. All the used methods are in the test file named "party_utest_1.cc" and are listed as follow:

1. SetCandidatesNames
2. GetCandidateName

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with an empty 2D vector of strings containing the names.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call SetCandidatesNames	A 2D vector of strings of candidates names {{"Foster", "Volz", "Pike"}, {"Green", "Xu", "Wang"}, {"Jacks", "Rosen"}, {"McClure", "Berg"}, {"Zheng", "Melvin"}, {"Peters"}}	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
2	Call GetCandidateName 13 times and pass in indexes of the parties and indexes of the candidates		Foster Volz Pike Green Xu Wang Jacks Rosen McClure Berg Zheng	Foster Volz Pike Green Xu Wang Jacks Rosen McClure Berg Zheng	

			Melvin Peters	Melvin Peters	

Postcondition(s) for Test: Names of the candidates are stored in a 2D vector of strings in the correct order and can be accessed anytime without having to worry about getting an out of range error of the vector.

—---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 3 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to test a setter and getter function in the class for the candidates' amounts. All the used methods are in the test file named "party_utest_2.cc" and are listed as follow:

1. SetAvailableCandidates
2. GetCandidatesCount
3. GetAvailableCandidates

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with two empty vectors of integers containing the numbers of candidates and the numbers of available candidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call SetAvailableCandidates	An integer vector of candidates {3, 3, 2, 2, 2, 1}	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
2	Call GetCandidatesCount six times and pass in indexes of the parties		3 3 2 2 2 1	3 3 2 2 2 1	
3	Call GetAvailableCandidates six times and pass in indexes of the parties		3 3 2 2 2 1	3 3 2 2 2 1	

Postcondition(s) for Test: Numbers of candidates and numbers of available candidates are stored in two vectors of integers in the correct order and can be accessed anytime without having to worry about getting an out of range error of the vector.

———————————————————————————————————————

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 4 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to test a setter and getter function in the class for the votes' amounts. All the used methods are in the test file named "party_utest_3.cc" and are listed as follow:

1. SetVotesAmounts
2. GetVotesAmount
3. GetInitialVotesAmount

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with two empty vectors of integers containing the votes amounts and the initial votes amounts.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call SetVotesAmounts	An integer vector of votes {5, 7, 3, 4, 2, 1}	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
2	Call GetVotesAmount six times and pass in indexes of the parties		5 7 3 4 2 1	5 7 3 4 2 1	
3	Call GetInitialVotesAmount six times and pass in indexes of a parties		5 7 3 4 2 1	5 7 3 4 2 1	

Postcondition(s) for Test: Numbers of the votes and numbers of initial votes are stored in two vectors of integers in the correct order and can be accessed anytime without having to worry about getting an out of range error of the vector.

—--

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 5 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to test a setter and getter function in the class for the votes' amounts. All the used methods are in the test file named "party_utest_4.cc" and are listed as follow:

1. SetAssignedSeats
2. GetAssignedSeats

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with an empty vector of integers containing the number of seats assigned to the parties.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call SetAssignedSeats	An integer vector of seats {0, 0, 0, 0, 0, 0}	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
2	Call GetAssignedSeats six times and pass in indexes of the parties		0 0 0 0 0 0	0 0 0 0 0 0	

Postcondition(s) for Test: Numbers of seats are stored in the vector of integers in the correct order and can be accessed anytime without having to worry about getting an out of range error of the vector.
——---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 6 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to do some tests on the number of candidates and make some modifications in the numbers to make sure they are handled correctly. All the used methods are in the test file named "party_utest_5.cc" and are listed as follow:

1. GetCandidatesCount
2. GetAvailableCandidates
3. ModifyAvailableCandidates

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and Party class instance has been created and the two vectors related to the candidates have had values {3, 3, 2, 2, 2, 1}.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call GetCandidatesCount six times and pass in indexes of the parties		3 3 2 2 2 1	3 3 2 2 2 1	
2	Call GetAvailableCandidates six times and pass in indexes of the parties		3 3 2 2 2 1	3 3 2 2 2 1	
3	Call ModifyAvailableCandidates six times and pass in indexes of the party and the amounts of candidates to be changed	Amount -1 Amount 0 Amount -1 Amount -2 Amount -1 Amount 0	No error is thrown due to vector being out of range or any reason	No error was thrown due to vector being out of range or any reason	
4	Call GetCandidatesCount six times and pass in indexes of the parties		3 3 2 2 2 1	3 3 2 2 2 1	

5	Call GetAvailableCandidates six times and pass in indexes of the parties		2 3 1 0 1 1	2 3 1 0 1 1	

Postcondition(s) for Test: Numbers of available candidates have been changed depending on the parties having been assigned seats or not.

—---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 7 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to do some tests on the number of seats and make some modifications in the numbers to make sure they are handled correctly. All the used methods are in the test file named "party_utest_6.cc" and are listed as follow:

1. GetAssignedSeats
2. ModifySeats
3. StoreFirstRoundSeats
4. GetFirstRoundSeats
5. GetAssignedSeats

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with a vector of integers containing the numbers of seats assigned to the parties {0, 0, 0, 0, 0, 0} and an empty vector of integers containing the number of seats after the whole number round.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call GetAssignedSeats six times and pass in an indexes of a parties		0 0 0 0 0 0	0 0 0 0 0 0	
2	Call ModifySeats six times and pass in the indexes of the parties and the amount of seats to be changed	Amount 1 Amount 1 Amount 0 Amount 2 Amount 0 Amount 1	No error is thrown due to vector being out of range or any reason	No error is thrown due to vector being out of range or any reason	
3	Call StoreFirstRoundSeats		No error is thrown due to vector being out of range or any reason, the said empty vector should now be accessible	No error is thrown due to vector being out of range or any reason, the said empty vector should now be accessible	

4	Call GetFirstRoundSeats six times and pass in indexes of the parties		1 1 0 2 0 1	1 1 0 2 0 1	
5	Call GetAssignedSeats six times and pass in indexes of the parties		1 1 0 2 0 1	1 1 0 2 0 1	
7	Call ModifySeats six times and pass in the indexes of the parties and the amount of seats to be changed	Amount 1 Amount 1 Amount 0 Amount 2 Amount 0 Amount 1	No error is thrown due to vector being out of range or any reason	No error is thrown due to vector being out of range or any reason	
8	Call GetFirstRoundSeats six times and pass in indexes of the parties		1 1 0 2 0 1	1 1 0 2 0 1	
9	Call GetAssignedSeats six times and pass in indexes of the parties		2 2 0 4 0 2	2 2 0 4 0 2	

Postcondition(s) for Test: Numbers of seats have been changed depending on the parties having been assigned seats or not.

—--

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 8 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to do some tests on the number of votes and make some modifications in the numbers to make sure they are handled correctly. All the used methods are in the test file named "party_utest_7.cc" and are listed as follow:

1. GetInitialVotesAmount
2. GetVotesAmount
3. ModifyVotes

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with two vectors of integers containing the numbers of votes of the parties and the initial number of votes of the parties, which are both {5, 7, 3, 4, 2, 1}.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call GetInitialVotesAmount six times and pass in an indexes of a parties		5 7 3 4 2 1	5 7 3 4 2 1	
2	Call GetVotesAmount six times and pass in the indexes of the parties		5 7 3 4 2 1	5 7 3 4 2 1	
3	Call ModifyVotes six times and pass in the indexes of the parties and the amount of seats to be changed	Amount -1 Amount -1 Amount 0 Amount -2 Amount 0 Amount -1	No error is thrown due to vector being out of range or any reason	No error is thrown due to vector being out of range or any reason	
4	Call GetInitialVotesAmount six times and pass in indexes of the parties		5 7 3 4 2 1	5 7 3 4 2 1	

5	Call GetVotesAmount six times and pass in indexes of the parties		4 6 3 2 2 0	4 6 3 2 2 0	

Postcondition(s) for Test: Numbers of votes have been changed after the whole number round and the process moves on to the remainder round.

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 9 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to do some tests on the number of additional seats that are assigned to the parties in the remainder round and make some modifications in the numbers to make sure they are handled correctly. All the used methods are in the test file named "party_utest_8.cc" and are listed as follow:

1. GetChangeInSeats
2. ModifyChangeInSeats

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with a vector of integers containing all zeros.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call GetChangeInSeats six times and pass in an indexes of a parties		0 0 0 0 0 0	0 0 0 0 0 0	
2	Call ModifyChangeInSeats six times and pass in the indexes of the parties and the amount of seats to be changed	Amount 0 Amount 1 Amount 2 Amount 1 Amount 1 Amount 0	No error is thrown due to vector being out of range or any reason	No error is thrown due to vector being out of range or any reason	
3	Call GetChangeInSeats six times and pass in indexes of the parties		0 1 2 1 1 0	0 1 2 1 1 0	

Postcondition(s) for Test: The numbers of additional seats of the parties are updated accordingly as the remainder round progresses.

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 10 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class Party. It is used to do some tests on the parties swapping functions to make sure they are handled correctly. All the used methods are in the test file named “party_utest_9.cc” and are listed as follow:

1. GetPartyName
2. GetCandidateName
3. GetCandidatesCount
4. GetAvailableCandidates
5. GetVotesAmount
6. SwapParties

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing information from an input file, the election has started processing, and a Party class instance has been created with a vector of strings containing parties names {"Democratic", "Republican", "New Wave", "Reform", "Green", "Independent"}, a 2D vector of strings containing the candidates names {{"Foster", "Volz", "Pike"}, {"Green", "Xu", "Wang"}, {"Jacks", "Rosen"}, {"McClure", "Berg"}, {"Zheng", "Melvin"}, {"Peters"}}, a vector of integers containing the number of available candidates {3, 3, 2, 2, 2, 1}, a vector of integers containings the numbers of candidates {3, 3, 2, 2, 2, 1}, a vector of integers containing the numbers of votes {5, 7, 3, 4, 2, 1}, a vector of integers containing the number of initial votes {5, 7, 3, 4, 2, 1}, a vector of integers containing the number of seats assigned {0, 0, 0, 0, 0, 0}, a vector of integers containing the numbers of seats assigned after the first round {0, 0, 0, 0, 0, 0}, a vector of integers containing the numbers of changes in seats {0, 0, 0, 0, 0, 0}.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Call GetPartyName and pass in an index	Index 2	New Wave	New Wave	
2	Call GetCandidateName twice and pass in the indexes	Indexes (2, 0) Indexes (2, 1)	Jacks Rosen	Jacks Rosen	
3	Call GetCandidatesCount and pass in an index	Index 2	2	2	
4	Call GetAvailableCandidat	Index 2	2	2	

	es and pass in an index				
5	Call GetVotesAmount and pass in an index	Index 2	3	3	
6	Call GetPartyName and pass in an index	Index 5	Independent	Independent	
7	Call GetCandidateName and pass in two indexes	Indexes (5, 0)	Peters	Peters	
8	Call GetCandidatesCount and pass in an index	Index 5	1	1	
9	Call GetAvailableCandidates and pass in an index	Index 5	1	1	
10	Call GetVotesAmount and pass in an index	Index 5	1	1	
11	Call SwapParties and pass in two indexes	Indexes (2, 5)	No error is thrown due to vector being out of range or any reason	No error is thrown due to vector being out of range or any reason	
12	Call GetPartyName and pass in an index	Index 5	New Wave	New Wave	
13	Call GetCandidateName twice and pass in the indexes	Indexes (5, 0) Indexes (5, 1)	Jacks Rosen	Jacks Rosen	
14	Call GetCandidatesCount and pass in an index	Index 5	2	2	

15	Call GetAvailableCandidates and pass in an index	Index 5	2	2	
16	Call GetVotesAmount and pass in an index	Index 5	3	3	
17	Call GetPartyName and pass in an index	Index 2	Independent	Independent	
18	Call GetCandidateName and pass in two indexes	Indexes (2, 0)	Peters	Peters	
19	Call GetCandidatesCount and pass in an index	Index 2	1	1	
20	Call GetAvailableCandidates and pass in an index	Index 2	1	1	
21	Call GetVotesAmount and pass in an index	Index 2	1	1	

Postcondition(s) for Test: The positions of two parties (not the above two, they are just examples) are swapped based on their remaining votes for later use in the remainder round.

—---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 11 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class CPL. It is used to call the constructor of CPL to make sure it has no issue creating an instance of CPL. All the used methods are in the test file named "cpl_utest_0.cc" and are listed as follow:

1. CPL()

Automated: Yes
Result: Pass
Preconditions for Test: The system has finished importing all information (except the ballots) from an input file, right after the election starts processing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Use EXPECT_EQ to check whether the \`cpl\` pointer is null		True	True	
2	Call CPL constructor and pass in necessary information	String of parties "Democratic, Republican, New Wave, Reform, Green, Independent" String of candidates "Foster, Volz, Pike\nGreen, Xu, Wang\nJacks, Rosen\nMcClure, Berg\nZheng, Melvin\nPeters" Total seats 3 Total ballots 9	No error is thrown due to any reason	No error is thrown due to any reason	

3	Use EXPECT_EQ to check whether the \`cpl\` pointer is null		False	False	

Postcondition(s) for Test: An instance of CPL is ready to be used to initialize an instance of Party class and to process the election.

—---

Test stage: Unit **Test Date:** 03/25/2023
Test Case ID#: 12 **Name of Tester(s):** Cuong Ha
Test Description: This is a unit test for class CPL. It is used to test if the CPL method can calculate the number of seats assigned to each party correctly.
Automated: No
Result: N/A
Note: This test was initially used to test whether or not a Party instance is feeded correct information within CPL class. However, the file could not be opened when we used the Google test framework, so we had to run the whole system (without using `gtest`) and let it read and calculate the results, then we manually checked the correctness. The table below shows the steps we did to calculate the results by hand. The following information is the correct information to be returned:

- Quota value: 3
- Votes received:
    - Democratic: 3
    - Republican: 3
    - New Wave: 2
    - Reform: 2
    - Green: 1
    - Independent: 0
- Vote per quota:
    - Democratic: 1
    - Republican: 0
    - New Wave: 0
    - Reform: 0
    - Green: 0
    - Independent: 0
- First seats allocation:
    - Democratic: 1
    - Republican: 0
    - New Wave: 0
    - Reform: 0
    - Green: 0
    - Independent: 0
- Remaining votes after first allocation:
    - Democratic: 0
    - Republican: 2
    - New Wave: 0
    - Reform: 2
    - Green: 1

    - Independent: 1
- Final seats allocation:
    - Democratic: 1
    - Republican: 1
    - New Wave: 0
    - Reform: 1
    - Green: 0
    - Independent: 0

Preconditions for Test: The system has finished importing all information (including the ballots counts) from an input file, right after the election starts processing.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Note**
1	Count ballots	1,,,,, 1,,,,, ,1,,,, ,,,,1, ,,,,,1 ,,,1,, ,,,1,, 1,,,,, ,1,,,,		Democratic: 3 Republican: 2 New Wave: 0 Reform: 2 Green: 1 Independent: 1	
2	Calculate quota	Ballots count: 9 Seats count: 3		3	
3	Calculate first seats allocation amount per party			Democratic: 1 Republican: 0 New Wave: 0 Reform: 0 Green: 0 Independent: 0	
4	Allocate seats and calculate remaining seats			2	
5	Calculate remaining votes			Democratic: 0 Republican: 2 New Wave: 0	

				Reform: 2 Green: 1 Independent: 1	
6	Rank the parties based on remaining votes			Republican: 2 Reform: 2 Green: 1 Independent: 1 New Wave: 1 Democratic: 0	
7	Allocate remaining seats to parties with most votes				
8	Get final seats allocation result			Republican: 1 Reform: 1 Green: 0 Independent: 0 New Wave: 0 Democratic: 1	

Postcondition(s) for Test: An instance of CPL is ready to be used to initialize an instance of Party class and to process the election.

—--

Test stage: Unit _X__ System ___ **Test Date: 3/25/23**
Test Case ID#: Ballot_utest_0 **Name of Tester(s): Andrew B**
Test Description:
Verify that Ballot public methods working correctly.
Ballot::Ballot()
Ballot::Ballot(line, id)
Ballot::GetCurrentVote()
Ballot::GetID()
Ballot::Increment()
Test is in file Ballot_utest_0.cc

Automated: Yes _X__ No ___
Result: Pass _X__ Fail ___
Preconditions for Test: N/A. All necessary ballot information is created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Setup - create ballot objects		NA	NA	
2	Test Ballot Constructor	Ballots 1,2 & 4	[0,3,1]	[0,3,1]	
3	Test Ballot Default Constructor	Ballot 3	-1	-1	
4	Test GetID	Ballots 1:4	0,1,-1,3	0,1,-1,3	
5	Test Increment	Ballots 1&2	1,2,3,-1,2,-1	1,2,3,-1,2,-1	Specifically tests the case where not all ballot fields are used
6					

Postcondition(s) for Test:
Ballots are created, incremented through all of their valid candidates, and interrogated for data at each step along the way.

Test stage: Unit _X__ System ___ **Test Date: 3/25/23**
Test Case ID#: Candidate_utest_0 **Name of Tester(s): Andrew B**
Test Description:
Verify that Candidate public methods are working correctly.
Candidate::Candidate()
Candidate::Candidate(name, party)
Candidate::GetVoteCount()
Candidate::Init()
Candidate::AssignBallot()
Candidate::RemoveBallot()
Candidate::GetVoteIDs()
Test is in file Candidate_utest_0.cc

Automated: Yes _X__ No ___
Result: Pass _X__ Fail ___
Preconditions for Test: Several ballots are created which can be assigned and moved between candidates.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Note**
1	Setup - create ballot objects and candidate objects		NA	NA	
2	Test Candidate Constructor	Candidate 1	John, R, 0	John, R, 0	
3	Test Candidate Default Constructor	Candidate 2	“”,””,0	“”,””,0	
4	Test Candidate Init	Candidate* candidates	John,R,Bernie,D	John,R,Bernie,D	
5	Test Assign Ballot	Candidate* candidates	1,2	1,2	Tests multiple types of ballots
6	Test Remove, Test ReAssign	Candidate* candidates	0,0,3	0,0,3	
7	Test Get Vote IDs	Candidate* candidates	“”,”1,2,0”	“”,”1,2,0”	Includes empty list

					edge case

Postcondition(s) for Test:
Candidates are created, ballots are assigned to them. The action of reallocating a candidate is simulated by removing ballots from one and reassigning to another. The full life cycle of a candidate object has been completed.

Test stage: Unit _X__ System ___ **Test Date: 3/25/23**
Test Case ID#: IR_utest_0 **Name of Tester(s): Andrew B**
Test Description:
Verify that the IR election constructor is working correctly.
IR::IR()
IR::GetCandidateName()
input_file.is_open()
IR::ValidCandidates()
IR::InputBallots()
Test is in file IR_utest_0.cc

Automated: Yes _X__ No ___
Result: Pass ___ Fail _X__
Preconditions for Test: Input header information is known

Note: The failure of this test is caused by a test framework issue, and not a bug in the code! System testing of the code clearly shows that when inputting test files, the system is able to use InputBallots without issues. However, when operating in the test environment the file stream fails to read data into the program and only blank strings are available for processing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Instantiate IR	4, `"Rosen(D),Kleinberg(R),Chou(I),Royce(L)", 6`	NA	NA	
2	Call GetCandidateName() for each candidate	candidate indices: [1,2,3,4]	Rosen;Kleinberg;Chou;Royce	Rosen;Kleinberg;Chou;Royce	
3	Test Set File Paths	Input_file	true	true	
4	Test Input Ballots	Election, IR_input_test_1.csv	4	0	

Postcondition(s) for Test:
An IR instance has been created with all the correct candidates in a member array. Ballot input file has been read, with ballots created and assigned appropriately.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Cli_preinput_1 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the command line with a file and date as arguments. Only entering one such argument should result in none being parsed; I should see Layout 1
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator string	“string” is the input parameter	Input parameter is ignored, display goes to election file selection (Layout 1)	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 1 (election file selection) awaiting election file information input.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Gui_preinput_1 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the command line with a file and date as arguments. Only entering one such argument should result in none being parsed; I should see Layout 1 GUI
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator_GUI string	“string” is the input parameter	Input parameter is ignored, display goes to election file selection (Layout 1)	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.

Postcondition(s) for Test:
System will be on Layout 1 (election file selection) awaiting election file information input.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Cli_preinput_2 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the command line with a file and date as arguments. Entering two arguments (the latter being a valid path to an existing formatted .csv electionfile) should result in the program launching to Layout 2, with displayed header information from the designated input file, awaiting user confirmation.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator 03-26-2023 cpl_input_test_0.csv	“03-26-2023” and “cpl_input_test_0.csv” are the input parameters, the latter is a valid file.	Election Information Page (Layout 2) is displayed detailing header information from file.	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_GUI_preinput_2 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the command line with a file and date as arguments. Entering two arguments (the latter being a valid path to an existing formatted .csv electionfile) should result in the program launching to Layout 2, with displayed header information from the designated input file, awaiting user confirmation, all on the GUI.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator_GUI 03-26-2023 cpl_input_test_0.csv	“03-26-2023” and “cpl_input_test_0.csv” are the input parameters, the latter is a valid file.	Election Information Page (Layout 2) is displayed detailing header information from file.	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.

Postcondition(s) for Test:
System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Cli_launch_1 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the makefile or the command line with no arguments (make run OR ./Tabulator)
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator OR make run	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 1 (Election File Selection) awaiting user confirmation.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Gui_launch_1 **Name of Tester(s): Brian Bianchi**
Test Description:
The application can be launched from the makefile or the command line with no arguments (make run_gui OR ./Tabulator_GUI)
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Note**
1	Enter into the command shell: ./Tabulator_GUI OR make run_gui	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.

Postcondition(s) for Test:
System will be on Layout 1 (Election File Selection) GUI awaiting user confirmation.

Test stage: Unit ___ System _X_ **Test Date: 3/26/2023**
Test Case ID#: Sys_Cli_input_1 **Name of Tester(s): Brian Bianchi**
Test Description:
Entering a valid date and IR election file should take the user to the Election Information page (Layout 2) and should present correct header information.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator OR make run	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	
2	Enter a date and press enter to confirm	“03-26-2023”	Election system should then prompt the user for a name of a file to load	Matched expectations.	
3	Enter a name of a valid IR election file and press enter	“IR_input_test_1.csv”	Election Information page displays with information matching election file header	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Gui_input_1 **Name of Tester(s): Brian Bianchi**

Test Description:

Entering a valid date and IR election file should take the user to the Election Information page (Layout 2) and should present correct header information.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator_GUI OR make run_gui	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.
2	Enter a date into the date box and a name of a valid IR election file into the file name box and press confirm	"03-26-2023", "IR_input_test_1.csv"		Matched expectations.	

Postcondition(s) for Test:

System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Cli_input_2 **Name of Tester(s): Brian Bianchi**
Test Description:
Entering a valid date and CPL election file should take the user to the Election Information page (Layout 2) and should present correct header information.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator OR make run	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	
2	Enter a date and press enter to confirm	“03-26-2023”	Election system should then prompt the user for a name of a file to load	Matched expectations.	
3	Enter a name of a valid CPL election file and press enter	“cpl_input_test_0.csv”	Election Information page displays with information matching election file header	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Gui_input_2 **Name of Tester(s): Brian Bianchi**

Test Description:

Entering a valid date and CPL election file should take the user to the Election Information page (Layout 2) and should present correct header information.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator_GUI OR make run_gui	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.
2	Enter a date into the date box and a name of a valid CPL election file into the file name box and press confirm	"03-26-2023", "cpl_input_test_0.csv"	Election Information page displays with information matching election file header	Matched expectations.	

Postcondition(s) for Test:

System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Cli_input_3 **Name of Tester(s): Brian Bianchi**
Test Description:
Entering a date and invalid/nonexistent election file should should print error and take user back to/keep at Layout 1 (Election File Selection) awaiting user re-entry of information.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator OR make run	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	
2	Enter a date and press enter to confirm	“03-26-2023”	Election system should then prompt the user for a name of a file to load	Matched expectations.	
3	Enter a name of an invalid election file and press enter	“test.csv”	Election File Selection page appears and an error is printed to the console	Matched expectations.	

Postcondition(s) for Test:
System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Gui_input_3 **Name of Tester(s): Brian Bianchi**

Test Description:

Entering a date and invalid/nonexistent election file should should print error and take user back to/keep at Layout 1 (Election File Selection) awaiting user re-entry of information.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter into the command shell: ./Tabulator_GUI OR make run_gui	N/A	Election File Selection page (Layout 1) appears and awaits user input	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.
2	Enter a date into the date box and a name of an invalid election file into the file name box and press confirm	“03-26-2023”, “test.csv”	Election File Selection page appears and an error is printed to the console	Matched expectations.	

Postcondition(s) for Test:

System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Cli_input_4 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to return from Layout 2 should take back to layout 1 for fresh file entry. Entering a file of a different type (CPL vs IR) should result in appropriate adjustment of header display on continuing back to Layout 2

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter any return input	Had loaded "cpl_input_test_0.csv", Entered "no"	Return to Election File Selection page	Matched expectations.	
2	Enter a date and press enter to confirm	"03-26-2023"	Election system should then prompt the user for a name of a file to load	Matched expectations.	
3	Enter a name of a valid election file of different election type	"IR_input_test_1.csv"	Election Information page displays with information matching election file header	Matched expectations.	

Postcondition(s) for Test:

System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Gui_input_4 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to return from Layout 2 should take back to layout 1 for fresh file entry. Entering a file of a different type (CPL vs IR) should result in appropriate adjustment of header display on continuing back to Layout 2

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Press"Back to File Selection"	Had loaded "cpl_input_test_0.csv"	Return to Election File Selection page	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.
2	Enter a date and enter a name of a valid election file of different election type	"03-26-2023", "IR_input_test_1.csv"	Election Information page displays with information matching election file header	Matched expectations.	

Postcondition(s) for Test:

System will be on Layout 2 (election information) awaiting user confirmation or return command.

Test Case ID#: Sys_Cli_confirm_1 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for an IR election

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on “Election Information” view page with an entered valid IR election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded “IR_input_test_1.csv”, Entered “Yes”	Results are tabulated and displayed for that election file on the Results page	Matched expectations.	

Postcondition(s) for Test:

System will be done running and awaiting user to close

Test Case ID#: Sys_Gui_confirm_1 **Name of Tester(s): Brian Bianchi**
Test Description:
Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for an IR election
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid IR election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Press "Confirm" button	Had loaded "IR_input_test_1.csv"	Results are tabulated and displayed for that election file on the Results page	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.

Postcondition(s) for Test:
System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_confirm_2 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for a CPL election

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid CPL election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded "cpl_input_test_0.csv", Entered "Yes"	Results are tabulated and displayed for that election file on the Results page	Matched expectations.	

Postcondition(s) for Test:

System will be done running and awaiting user to close

Test Case ID#: Sys_Gui_confirm_2 **Name of Tester(s): Brian Bianchi**
Test Description:
Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for an CPL election
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator_GUI built using command: make gui. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid CPL election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Press "Confirm" button	Had loaded "cpl_input_test_0.csv"	Results are tabulated and displayed for that election file on the Results page	Matched expectations.	The GUI cannot be built without GTKMM dev on the system; this test was done on a pre-built executable from a machine with GTKMM dev.

Postcondition(s) for Test:
System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_confirm_3 **Name of Tester(s): Brian Bianchi**
Test Description:
Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for a IR election AND a file with 100,000 or more ballots should not take more than 4 minutes to process.
Automated: Yes ___ No _X_
Result: Pass _X_ Fail ___
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot IR election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded "IR_input_t test_3.csv", Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes	Matched expectations - less than 5 seconds calculation time!	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:
System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_confirm_4 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for a CPL election AND a file with 100,000 or more ballots should not take more than 4 minutes to process.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot CPL election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded "cpl_input_test_7.csv", Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes	Matched expectations - less than 0.71 second calculation time!	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:

System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_audit_1 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for an IR election AND a file with 100,000 or more ballots should not take more than 4 minutes to process. An audit file detailing the processing of the election should also be generated, named for the entered date.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot IR election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded "IR_input_t test_3.csv", "3-26-2023 " Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes. An audit file detailing the order of processing of the election is generated in an audit file including the passed in date in its name.	Matched expectations - Audit file generated!	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:

System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_audit_2 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for a CPL election AND a file with 100,000 or more ballots should not take more than 4 minutes to process. An audit file detailing the processing of the election should also be generated, named for the entered date.

Automated: Yes ___ No _X_

Result: Pass _X_ Fail ___

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot CPL election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Enter confirmation command	Had loaded "cpl_input_test_7.csv", "3-26-2023" Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes. An audit file detailing the order of processing of the election is generated in an audit file including the passed in date in its name.	Matched expectations - Audit file generated!	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:

System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_audit_3 **Name of Tester(s): Brian Bianchi**
Test Description:
Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for an IR election AND a file with 100,000 or more ballots should not take more than 4 minutes to process. An audit file detailing the processing of the election should also be generated, named for the entered date.
Automated: Yes ___ No _X_
Result: Pass ___ Fail _X_
Preconditions for Test:
In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot IR election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Chose a date with slashes in the title. Enter confirmation command	Had loaded "IR_input_t test_3.csv", "3/26/2023" Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes. An audit file detailing the order of processing of the election is generated in an audit file including the passed in date in its name.	Everything worked fine except that no audit file was generated.	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:
System will be done running and awaiting user to close

Test Case ID#: Sys_Cli_audit_4 **Name of Tester(s): Brian Bianchi**

Test Description:

Choosing to confirm from Layout 2 should result in matching tabulated file results displayed (Layout 4) for a CPL election AND a file with 100,000 or more ballots should not take more than 4 minutes to process. An audit file detailing the processing of the election should also be generated, named for the entered date.

Automated: Yes ___ No _X_

Result: Pass ___ Fail _X_

Preconditions for Test:

In a shell/command line, navigated to and in Project1 folder, Tabulator built using command: make. Must have a valid election .csv file to input and it must be in either Project1 or Project1/testing. Must be on "Election Information" view page with an entered valid 100,000 or more ballot CPL election file displayed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note
1	Chose a date with slashes in the title. Enter confirmation command	Had loaded "cpl_input_test_7.csv", "3/26/2023" Entered "Yes"	Results are tabulated and displayed for that election file on the Results page in under 4 minutes. An audit file detailing the order of processing of the election is generated in an audit file including the passed in date in its name.	Everything worked fine except that no audit file was generated.	Same subroutines used in GUI, testing this with that would be unnecessarily redundant.

Postcondition(s) for Test:

System will be done running and awaiting user to close