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 notes for you and your team members.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/22/23**

Test Case ID#: Audit_Constructor **Name(s) of Testers: Liam O'Neil**

Test Description:

Test the Audit constructor's file creation given a file path.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
testAudit() method stored in src/AuditTest.java

Results: Pass

Preconditions for Test:
A String containing the file path and the instantiation of an Audit object.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Call Audit constructor with parameter "audit.txt" to create an audit file named "audit.txt".	String "audit.txt"	Created file "audit.txt" in Project1/src	Created file "audit.txt" in Project1/src	
3					
4					

Post condition(s) for Test:

No change in system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: Audit_appendString **Name(s) of Testers: Liam O'Neil**

Test Description:
Test that appendString() correctly appends to a file.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
appendStringTest() stored in src/AuditTest.java

Results: Pass

Preconditions for Test:
A String containing the text to append to the file, and the initialization of an Audit object.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Call appendString() with input parameter "test text" and "".	String "test text", empty String "".	"test text\n" appended to Audit file.	"test text\n" appended to Audit file.	
3					
4					

Post condition(s) for Test:

No change in system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_simulate_IR **Name(s) of Testers: Liam O'Neil**

Test Description:
Testing the simulate() method for IR cases.
Note that the data used for these tests is taken from the listed CSV test files in the /testing directory. These files are not used as input.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
simulateIRTest() stored in src/FileParserTest.java

Results: Pass

Preconditions for Test:
String containing the expected output, initialized FileParser object with IRCandidate List, Ballot List data present in /testing/

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call simulate() with all necessary IR related objects manually initialized with the data in /testing/IRV.csv	Ballots and candidates listed in IRV.csv	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates "Rosen (D)" as the winner.	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates "Rosen (D)" as the winner.	
2	Call simulate() with all necessary IR related objects manually initialized with the data in IRV2.csv	Ballots and candidates listed in IRV2.csv	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates either "Grant (I)" or "Chou (I)" as the winner.	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates "Grant (I)" as the winner.	The two final remaining candidates are expected to be tied, thus it is tested that the indicated winner is not one of the 4 other candidates.
3	Call simulate() with all necessary IR related objects manually initialized with the data in IRV3.csv	Ballots and candidates listed in IRV3.csv	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates either "Wilt (D)" or "Chou (I)" as the winner.	The expected output matches the data in the electionInfo Election attribute. The getResults() method indicates "Wilt (D)" as the winner.	The two final remaining candidates are expected to be tied, thus it is tested that the indicated winner is not one of the 6 other candidates.

Post condition(s) for Test:

No change in system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_simulate_CPL **Name(s) of Testers: Liam O'Neil**

Test Description:
Testing the simulate() method for CPL cases.
Note that the data used for these tests is taken from the listed CSV test files in the /testing directory. These files are not used as input.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
simulateCPLTest() stored in src/FileParserTest.java

Automated: yes

Results: Pass

Preconditions for Test:
String containing the expected output, initialized FileParser object with IRCandidate List, Ballot List data present in /testing/

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call simulate() with all necessary CPL related objects manually initialized with the data present in CPL.csv	Ballots and parties listed in CPL.csv	The expected output matches the data in the electionInfo Election attribute.	The expected output matches the data in the electionInfo Election attribute.	
2	Call simulate() with all necessary CPL related objects manually initialized with the data present in CPL2.csv	Ballots and parties listed in CPL2.csv	The expected output matches the data in the electionInfo Election attribute.	The expected output matches the data in the electionInfo Election attribute.	
3	Call simulate() with all necessary CPL related objects manually initialized with the data present in CPL3.csv	Ballots and parties listed in CPL3.csv	The expected output matches the data in the electionInfo Election attribute.	The expected output matches the data in the electionInfo Election attribute.	

Post condition(s) for Test:

No change in system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: FileParser_constructor

Name(s) of Testers: Bilal Osman

Test Description:
Testing that the FileParser constructor stores the input CSV file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
FileParserTest() stored in src/FileParserTest.java

Automated: yes

Results: Pass

Preconditions for Test:
Initialization of FileParser objects with the input CSV file names.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1					
2	Call FileParser() constructor with “CPL.csv” parameter	n/a	The file is retrievable by the getFile() method of FileParser	The file is retrievable by the getFile() method of FileParser.	
3	Call FileParser() constructor with “IRV.csv” parameter	n/a	The file is retrievable by the getFile() method of FileParser	The file is retrievable by the getFile() method of FileParser	
4					

Post condition(s) for Test:

No change in system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_getFile **Name(s) of Testers: Bilal Osman**

Test Description:

Testing that getFile() returns the expected file object.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

getFileTest() in src/FileParserTest.java

Automated: yes

Results: Pass

Preconditions for Test:

Initialized FileParser object with input filename and File object initialized with input filename. Exists a file “temp.csv” in working directory.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Call getFile() from the FileParser object	filename “temp.csv” as input parameter	FileParser.getFile() and File object initialized with “temp.csv” are equal.	FileParser.getFile() and File object initialized with “temp.csv” are equal.	
3					
4					

Post condition(s) for Test:

No change to system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_createBallot **Name(s) of Testers: Bilal Osman**

Test Description:
Testing that the createBallot() method of FileParser creates and stores ballot information correctly.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
createBallotTest() in src/FileParserTest.java

Results: Pass

Preconditions for Test:
FileParser object initialized with ‘temp.csv’. Exists a file “temp.csv” in working directory.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Call createBallot(ranks,candidates)	ranks = (“1”,”2”,”3”) candidates = (“cand1”,”cand2”,”cand3”);	createBallots(ranks,candidates) = manually initialized ballot using the same candidates/ranks	createBallots(ranks,candidates) = manually initialized ballot using the same candidates/ranks	
3					
4					

Post condition(s) for Test:

No change to system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_makeCandidates **Name(s) of Testers: Bilal Osman**

Test Description:
Testing that the makeCandidates() method of FileParser correctly creates candidate object

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
makeCandidateTest() in src/FileParserTest.java

Results: Pass

Preconditions for Test:
"temp.csv" file in current working directory.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Call makeCandidates("expected")	n/a	makeCandidates("expected") = manually created candidate object	makeCandidates("expected") = manually created candidate object	
3					
4					

Post condition(s) for Test:

No change to system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: FileParser_makeParties **Name(s) of Testers: Bilal Osman**

Test Description:
Testing that the makeParties() method of FileParser creates PoliticalParty objects correctly.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
makePartiesTest() in src/FileParserTest.java

Results: Pass

Preconditions for Test:
"temp.csv" in working directory.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Call makeParties("expected")	n/a	makeParties("expected") = manually created PoliticalParty object	makeParties("expected") = manually created PoliticalParty object	
3					
4					

Post condition(s) for Test:

No change to system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: Ballot_Constructor **Name(s) of Testers: Bilal Osman**

Test Description:
Tests the constructor of the Ballot class and see if passes in the correct Hashmap of candidate names to rankings

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
BallotTest() method stored in src/BallotTest.java

Automated: yes

Results: Pass

Preconditions for Test: A hashmap containing a tuple of candidate names and an integer ranking

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	create a hashmap with three key and value pairs of a string name and an integer ranking, then make a ballot object with the same hashmap	Map<String, Integer> maptest and Ballot bt	Map<String, Integer> maptest	bt.getVote() = maptest	
3					
4					

Post condition(s) for Test:

No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: Get_Vote_Test **Name(s) of Testers: Bilal Osman**

Test Description:
Tests the method getVote() to see if it returns the correct hashmap even when the other hashmap changed

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
GetVoteTest() method stored in src/BallotTest.java

Automated: yes

Results: Pass

Preconditions for Test: none

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	create two hashmap with three key and value pairs of a string name and an integer ranking, then make a ballot object with the one of the hashmaps	Map<String, Integer> maptest and Ballot bt	Map<String, Integer> maptest	bt.getVote() = maptest	
3	remove an element from the second hashmap and compare using assertNotEquals	Map<String, Integer> maptest2 and Ballot bt	Map<String, Integer> maptes2	bt.getVote() != maptest2	
4					

Post condition(s) for Test:

No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: Get_Vote_from_Candidate **Name(s) of Testers: Bilal Osman**

Test Description:
Tests the method getVoteforCandidate() to see if the method returns the correct ranking from the hashmap

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
GetVoteForCandidateTest() method stored in src/BallotTest.java

Automated: yes

Results: Pass

Preconditions for Test: none

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a hashmap with two elements. Test passing in the name of the first element	Integer(100) and rating from first element	100	100	
3	Test passing in the name of the second element	Integer(200) and rating from second element	200	200	
4	Test passing in the name of an element that doesn't exist	Integer(300)	0	0	

Post condition(s) for Test:

No change to system state.

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#:Get_Candidate_from_Preference

Name(s) of Testers: Bilal Osman

Test Description:
Tests the method GetCandidateFromPreference to see if it returns the correct candidate that has the preference rating passed in

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
GetCandidateFromPreferenceTest() method stored in src/BallotTest.java

Automated: yes

Results: Pass

Preconditions for Test: none

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create HashMap of candidate names and preferences and make a Ballot object. Call getCandidateFromPreference() and look at rating of 100	Ballot bt = new Ballot({(cand1,1), (cand2,2))	cand1	cand1	
3	Same test but with 200	Ballot bt = new Ballot({(cand1,1), (cand2,2))	cand2	cand2	
4	Same test but with 300 which dosent exist in the ballot	Ballot bt = new Ballot({(cand1,1), (cand2,2))	null	null	

Post condition(s) for Test:

No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty Constructor **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description: Tests functionality of Political Party constructor to check if the attributes are properly initialized

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used. testPoliticalParty() method in PoliticalPartyTest.java

Results: Pass

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Create an ordered queue of strings containing candidate names. Create the name of a politicalParty, and call the constructor.	Queue<String> candidateNames, partyName	partyName: "Democrats" candidateNames: "biden", "kamala" partySeats: 0 partyVotes: 0 remainder: 0	partyName: "Democrats" candidateNames: "biden", "kamala" partySeats: 0 partyVotes: 0 remainder: 0	
3					
4					

Post condition(s) for Test:
No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: IRCandidate Constructor **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:

Tests functionality of IRCandidate constructor to check if the attributes are properly initialized

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

Automated: yes

Results: Pass

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Creates a candidate containing the name of the candidate, the number of votes, and the votes' percentage.	String name	name : "biden" votes : 0 percentVotes : 0	name : "biden" votes : 0 percentVotes : 0	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty getPartyName **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description: Testing a getter function of an attribute, and comparing the string output.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used. getPartyNameTest() method in PoliticalPartyTest.java

Results: Pass

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	check string partyName	String partyName	partyName: "Democrats"	partyName: "Democrats"	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty getCandidateNames **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:
Testing the getter function for the names of the candidates in party, in an ordered queue

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getCandidateNamesTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: none

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the functionality of candidateNames getter functon by checking if it returns the correct queue that it was intialized with in the constructor.	Queue<String> candidateNames	candidateName: "Biden", "Kamala"	candidateName: "Biden", "Kamala"	
2					
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: PoliticalParty getPartySeats

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

Test Description:
Testing the getter function of the partySeats attribute, and comparing the int output.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getPartySeatsTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the functionality of partySeats getter functon by checking if it returns the correct value that it was intialized/set with.	int partySeats	partySeats: 2	partySeats: 2	
2					
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty getPartyVotes **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:
Testing the getter function of the partyVotes attribute, and comparing the int output.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getPartyVotesTest() method in PoliticalPartyTest.java

Results: Pass

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the functionality of partyVotes getter functon by checking if it returns the correct value that it was intialized/set with.	int partyVotes	partyVotes: 3	partyVotes: 3	
2					
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty setPartySeats **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:
Checks if the partySeats attribute is properly assigned in the setter function.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
setPartySeatsTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of partySeats setter function. Checks if the attribute was properly assigned.	int partySeats	partySeats: 2	partySeats: 2	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty setPartyVotes **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:
Checks if the partyVotes attribute is properly assigned in the setter function.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
setPartyVotesTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of partyVotes setter function. Checks if the attribute was properly assigned.	int partyVotes	partyVotes: 3	partyVotes: 3	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty setRemainderVotesTest **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:
Checks if the votes attribute is properly assigned in the setter function.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
setRemainderVotesTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Check the functionality of RemainderVotes setter function. Checks if the attribute was properly assigned.	int votes	votes: 4	votes: 4	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: PoliticalParty getRemainderTest **Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu**

Test Description:

Testing the getter function of the remainder attribute, and comparing the int output.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getRemainderTest() method in PoliticalPartyTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of RemainderVotes getter functon by checking if it returns the correct value that it was intialized/set with.	int votes	votes: 4	votes: 4	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: System	**Test Date: 3/28/23**
Test Case ID#: System_CPL_1	**Name(s) of Testers: Liam O'Neil**
Test Description: Testing the overall functionality of the system for a CPL case.	
Automated: no	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** Manual test using "CPL.csv" in /testing/
Results: Pass	

Preconditions for Test:
Access to "CPL.csv" in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using "java FileParser /testing/CPL.csv"	CPL.csv	Correct allocation of seats displayed to user and correct data present in the audit file.	Correct allocation of seats displayed to user and correct data present in the audit file.	

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “CPL.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_CPL_2 **Name(s) of Testers: Liam O’Neil**

Test Description:
Testing the overall functionality of the system for a CPL case.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using “CPL2.csv” in /testing/

Automated: no

Results: Pass

Preconditions for Test:
Access to “CPL2.csv” in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using “java FileParser /testing/CPL2.csv”	CPL2.csv	Correct allocation of seats displayed to user and correct data present in the audit file.	Correct allocation of seats displayed to user and correct data present in the audit file.	

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “CPL2.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_CPL_3 **Name(s) of Testers: Liam O'Neil**

Test Description:
Testing the overall functionality of the system for a CPL case.

Automated: no

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using "CPL3.csv" in /testing/

Results: Pass

Preconditions for Test:
Access to "CPL3.csv" in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using "java FileParser /testing/CPL3.csv"	CPL3.csv	Correct allocation of seats displayed to user and correct data present in the audit file.	Correct allocation of seats displayed to user and correct data present in the audit file.	

Post condition(s) for Test:

Modified "audit.txt" includes audit information for the election data in "CPL3.csv".

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_CPL_4 **Name(s) of Testers: Liam O'Neil**

Test Description:
Testing the overall functionality of the system for a CPL case in which two parties have the same remainder votes following the first allocation of seats.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using "CPL_tiedRemainder.csv" in /testing/

Automated: no

Results: Fail

Preconditions for Test:
Access to "CPL_tiedRemainder.csv" in /testing directory

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Run program using "java FileParser /testing/CPL_tiedRemainder.csv"	CPL_tiedRemainder.csv	Correct allocation of seats displayed to user and correct data present in the audit file. The single available seat should be awarded randomly in the second allocation of seats.	The single available seat is not awarded randomly to one of the two parties. "Republican" is always award the seat.	Bug #1 documented in buglist.txt.

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “CPL_tiedRemainder.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_IR_1 **Name(s) of Testers: Liam O’Neil**

Test Description:
Testing the overall functionality of the system for an IR case.

Automated: no

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using “IRV.csv” in /testing/

Results: Pass

Preconditions for Test:
Access to “IRV.csv” in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using “java FileParser /testing/IRV.csv”	IRV.csv	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is “Rosen (D)”.	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is “Rosen (D)”.	

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “IRV.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_IR_2 **Name(s) of Testers: Liam O’Neil**

Test Description:

Testing the overall functionality of the system for an IR case.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

Manual test using “IRV2.csv” in /testing/

Automated: no

Results: Pass

Preconditions for Test:

Access to “IRV2.csv” in /testing directory

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run program using “java FileParser /testing/IRV2.csv”	IRV2.csv	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is “Grant (I)” or “Chou (I)”	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is “Chou (I)”.	The election will result in Chou (I) and Grant (I) being tied in the final count. One of the two is chosen randomly as the winner.

Post condition(s) for Test:

Modified "audit.txt" includes audit information for the election data in "IRV2.csv".

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_IR_3 **Name(s) of Testers: Liam O'Neil**

Test Description:

Testing the overall functionality of the system for an IR case.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

Manual test using "IRV3.csv" in /testing/

Automated: no

Results: Pass

Preconditions for Test:

Access to "IRV3.csv" in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using "java FileParser /testing/IRV3.csv"	IRV3.csv	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is "Wilt (D)" or "Chou (I)"	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is "Wilt (D)".	Wilt (D) and Chou (I) will be tied in the final count. Winner is chosen randomly.

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “IRV3.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_IR_4 **Name(s) of Testers: Liam O’Neil**

Test Description:
Testing the overall functionality of the system for an IR case in which two candidates are tied.

Automated: no

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using “IR_tie.csv” in /testing/

Results: Pass

Preconditions for Test:
Access to “IR_tie.csv” in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using “java FileParser /testing/IR_tie.csv”	IR_tie.csv	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is chosen randomly between the two candidates.	Type of election and winner correctly displayed to user. Audit file contains election info and correct data for each count. Winner is chosen randomly between the two candidates.	

Post condition(s) for Test:

Modified “audit.txt” includes audit information for the election data in “IR_tie.csv”.

Project Name: Project 1: Voting System **Team#14**

Test Stage: System **Test Date: 3/28/23**

Test Case ID#: System_IR_5 **Name(s) of Testers: Liam O'Neil**

Test Description:

Testing the overall functionality of the system for an IR case in which no second preferences are present on the ballots. Ensuring the percent of votes for each candidate is accurate and that ballots with no 2nd preference are removed from the total vote count when necessary.

Automated: no

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using "IR_no2ndPref.csv" in /testing/

Results: Pass

Preconditions for Test:
Access to "IR_no2ndPref.csv" in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using "java FileParser /testing/IR_no2ndPref.csv"	IR_no2ndPref.csv	Percent of votes for each candidate in count 2 is accurate. Chou (I): ~43 Rosen (D): ~57	Percent of votes for each candidate in count 2 is accurate. Chou (I): ~43 Rosen (D): ~57	

Post condition(s) for Test:

Modified "audit.txt" includes audit information for the election data in "IR_no2ndPref.csv".

Project Name: Project 1: Voting System **Team#14**

Test Stage: System

Test Date: 3/28/23

Test Case ID#: System_IR_6

Name(s) of Testers: Liam O'Neil

Test Description:
Testing the overall functionality of the system for an IR case with one candidate.

Automated: no

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Manual test using "IR_1cand.csv" in /testing/

Results: Pass

Preconditions for Test:
Access to "IR_1cand.csv" in /testing directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program using "java FileParser /testing/IR_1cand.csv"	IR_1cand.csv	Program performs a single count of the votes before declaring the winner.	Program performs a single count of the votes before declaring the winner.	

Post condition(s) for Test:

Modified "audit.txt" includes audit information for the election data in "IR_1cand.csv".

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: IRCandidate getName

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

Test Description:

Testing the getter function of the name attribute, and comparing the String output.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getNameTest() method in IRCandidateTest.java

Automated: yes

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of name getter functon by checking if it returns the correct value that it was intialized/set with.	String name	name: "biden"	name: "biden"	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit **Test Date: 3/26/23**

Test Case ID#: IRCandidate getVotes
Test Description:
Testing the getter function of the votes attribute, and comparing the int output.

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getVotesTest() method in IRCandidateTest.java

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of votes getter functon by checking if it returns the correct value that it was intialized/set with.	int votes	votes: 2	votes: 2	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: IRCandidate getPercent

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

Test Description:
Testing the getter function of the percent attribute, and comparing the double output. We use Delta variable to check if the actual result is within a specified tolerance, as Java does not allow the exact comparison of two double values.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
getPercentTest() method in IRCandidateTest.java

Results: Pass

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Check the functionality of percent getter functon by checking if it returns the correct value that it was intialized/set with. Compare using delta as tolerance.	double percent double DELTA	percent: 25.5 DELTA: 1e-15	percent: 25.5 DELTA: 1e-15	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System

Team#14

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: IRCandidate setVotes
Test Description:
Checks if the votes attribute is properly assigned in the setter function.

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
setVotesTest() method in IRCandidateTest.java

Results: Pass

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Check the functionality of votes setter function. Checks if the attribute was properly assigned.	int votes	votes: 2	votes: 2	
3					
4					

Post condition(s) for Test: No change in system state

Project Name: Project 1: Voting System **Team#14**

Test Stage: Unit

Test Date: 3/26/23

Test Case ID#: IRCandidate setPercent
Test Description:
Checks if the percent attribute is properly assigned in the

Name(s) of Testers: Soorya Sundravel, Hyehwan Ryu

setter function. We use Delta variable to check if the actual result is within a specified tolerance, as Java does not allow the exact comparison of two double values.

Automated: yes

Indicate where are you storing the tests (what file) and the name of the method/functions being used. setPercentTest() method in IRCandidateTest.java

Results: Pass

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Check the functionality of percent setter function. Checks if the attribute was properly assigned. Compare using delta as tolerance.	double percent double DELTA	percent: 25.5 DELTA: 1e-15	percent: 25.5 DELTA: 1e-15	
3					
4					

Post condition(s) for Test: No change in system state