Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# CPLBallot_Constructor
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that the CPLBallot constructor is declared and initialized correctly

Automated: yes_X_ **no__**
Results: **Pass_X_** **Fail__**

Preconditions for Test:
CPLBallot cplballot declared and initialized

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert that the constructor works as intended.	CPLBallot cplballot	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# CPLBallot_getPartyChoice
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that getPartyChoice correctly returns the ballot's party choice

Automated: **yes_X_** **no__**
Results: **Pass_X_** **Fail__**

Preconditions for Test:
CPLBallot cplballot declared and initialized

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert that the getPartyChoice() method returns the party choice	cplballot.getPartyChoice()	CPLParty party (Specified in the test)	CPLParty party	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# CPLFileProcessor_Constructor
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that CPLFileProcessor constructor is declared and initialized correctly

Automated: yes_X_ **no__**
Results: **Pass_X_** **Fail__**

Preconditions for Test:
CPLFileProcessor processor declared and initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert that the constructor works as intended.	CPLFileProcessor processor	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# CPLFileProcessor_processFile_GOOD
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that processFile correctly processes the data in a valid CPL election file

---

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
CPLFileProcessor processor declared and initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call processFile and store information in new CPLElection object good+_election	testCPL.csv			
2	Check that election data is correct in good_election	getNumSeats(), getNumBallots(), getNumVoteables()	Equal to expected value	Equal to expected value	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# CPLFileProcessor_processFile_BAD
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that processFile correctly processes the data in a nonexistent file

Automated: **yes_X_** **no__**
Results: **Pass_X_** **Fail__**

Preconditions for Test:
CPLFileProcessor processor declared and initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call processFile on null_file	File null_file	NullPointerException	NullPointerException	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# CPLParty_Constructor
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that the CPLParty constructor is declared and initialized correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
CPLBallot cplballot declared and initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert that the constructor works as intended.	CPLParty party_empty , CPLParty party_2	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# CPLParty_Constructor
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that the CPLParty constructor is declared and initialized correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
CPLParty party_empty and CPLParty party_2 declared and initialized

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert that the constructor works as intended.	CPLParty party_empty , CPLParty party_2	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# CPLParty_Getters
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that the CPLParty getters all work correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
CPLParty party_empty and CPLParty party_2 declared and initialized

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert that getPartyCandidates works correctly	CPLParty party_empty, CPLParty party_2	NotNull	NotNull	
2	Assert that getNumPartyCandidates works correctly	CPLParty party_empty, CPLParty party_2	NotNull	NotNull	
3	Assert that getTopPartyCandidates works correctly	CPLParty party_empty, CPLParty party_2	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# CPLParty_Setters
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:

Tests that the CPLParty setters all work correctly

Automated: yes_X_ **no__**
Results: Pass_X_ **Fail__**

Preconditions for Test:

CPLParty party_empty and CPLParty party_2 declared and initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert that setNumSeatsAllottedFirst works correctly with valid data	CPLParty party_empty	Equal to expected value	Equal to expected value	
2	Assert that setNumSeatsAllottedFirst throws exception with invalid input	CPLParty party_empty	Thrown exception	Thrown exception	
3	Assert that setNumSeatsAllottedSecond works correctly	CPLParty party_empty	Equal to expected value	Equal to expected value	
4	Assert that setNumSeatsAllottedSecond throws exception with invalid input	CPLParty party_empty	Thrown exception	Thrown exception	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRBallot_Constructor
Name(s) of Testers: Ethan Loukusa, Gideon Tan

Test Description:
Tests that the IRBallot constructor is declared and initialized correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
LinkedList<IRCandidate> cands declared and initialized.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create IRBallot constrTestBallot				
2	Assert that constrTestBallot is not null	IRBallot constrTestBallot	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRBallot_redistributeVote
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that the redistributeVote method works for all cases.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
LinkedList<IRCandidate> cands declared and initialized.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create eliminated CandidatesQueue and add bob_cand to it.				
2	Call Redistribute Vote on ballot with eliminatedCandidatesQueue				
3	Assert that the length of ballots.voteables hasn't changed	ballots.candidatesQueue .length	3	3	
4	Assert that each candidate in ballot is what it should be	ballots.candidatesQueue	Alice_cand Bob_cand Charles_cand	Alice_cand Bob_cand Charles_cand	
5	Add alice_cand to eliminatedCandidatesQueue				

	Call Redistribute Vote on ballot with eliminatedCandidatesQueue				
6	Assert that the length of ballots.voteables has changed	ballots.candidatesQueue.length	1	1	
7	Assert that each candidate in ballot is what it should be	ballots.candidatesQueue	Charles_cand	Charles_cand	
8	Call Redistribute Vote on ballot with eliminatedCandidatesQueue				
9	Assert that the length of ballots.voteables has changed	ballots.candidatesQueue.length	0	0	
10	Call Redistribute Vote on ballot with eliminatedCandidatesQueue				
11	Assert that the length of ballots.voteables hasn’t changed	ballots.candidatesQueue.length	0	0	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRCandidate_Constructor
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that the IRCandidate constructor is declared and initialized correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create IRCandidate constrTestCandidate				
2	Assert that constrTestCandidate is not null	IRBallot constrTestCandidate	NotNull	NotNull	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRCandidate_Update_Vote_Count_History
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that the VoteCountHistory Array is correctly updated.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

IRCandidate “candidate” is created, initialized.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Increment candidate votes by 2				
2	Assert that the voteCountHistory has been updated correctly	candidate.voteCountHistory(0)	2	2	
3	Increment candidate votes by 0				
4	Assert that the voteCountHistory has been updated correctly	candidate.voteCountHistory(1)	2	2	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRFileProcessor_Constructor
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that a File Processor is initialized correctly.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create and initialize IRFileProcessor processor				
2	Assert that processor is not Null	processor	Not Null	Not Null	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# IRFileProcessor_ProcessFile_GOOD
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that a good file is processed correctly by an IRFileProcessor

---

Automated: yes_X_ **no__**
Results: Pass_X_ **Fail__**

Preconditions for Test:

An IRFileProcessor object "processor" is initialized with a good file as input.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call processor. processFile() and store the results in an IRElection object				
2	Assert that the IRElection object has the correct number of ballots	IRElection.numBallots	15	15	
3	Assert that the IRElection object has the correct number of candidates	IRElection.numCandidates	4	4	
4	Assert that all of the ballots are initialized	Ballot ballot in IRElection.ballots	Not Null	NotNull	
5	Assert that all the ballots have at least one candidate in them	Ballot ballot in IRElection.ballots	Size > 0 == true	Size > 0 == true	

6	Assert that all of the candidates are initialized	IRCandidate candidate in IRElection.candidates	Not Null	NotNull	
7	Assert that all the candidates names are all created	IRCandidate candidate.name in IRElection.candidates	Not Null	Not Null	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IRFileProcessor_ProcessFile_NOBALLOTS
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that a file with 0 ballots is processed correctly by an IRFileProcessor.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
An IRFileProcessor object "processor" is created

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call processor. processFile() on an input file with not ballots and store the results in an IRElection object				
2	Assert that the election object was created and initialized	IRElection election	Not Null	Not Null	
3	Assert that the IRElection object has the correct number of ballots	IRElection.numBallots	0	0	
4	Assert that the IRElection object has the correct number of candidates	IRElection.numCandidates	4	4	
5	Assert that all of the ballots are	Ballot ballot in	Not Null	NotNull	

	initialized (Shouldn’t actually ever reach this assertion)	IRElection.ballots			
6	Assert that all the ballots have at least one candidate in them (Shouldn’t actually ever reach this assertion)	Ballot ballot in IRElection.ballots	Size > 0 == true	Size > 0 == true	
	Assert that all of the candidates are initialized	IRCandidate candidate in IRElection.candidates	Not Null	NotNull	
	Assert that all the candidates names are all created	IRCandidate candidate.name in IRElection.candidates	Not Null	Not Null	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# IRFileProcessor_ProcessFile_BAD
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests that IRProcessor.processFile() handles bad input files.

Automated: yes_X_ **no__**
Results: Pass_X_ **Fail__**

Preconditions for Test:

An IRFileProcessor object "processor" is created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call processor. processFile() on nonexistentFile				
2	Assert that step one caused an Exception to be thrown	processor.processFile()	Illegal Argument Exception thrown	Illegal Argument Exception thrown	
3	Call processor. processFile() on a null file pointer				
4	Assert that step three caused an Exception to be thrown	processor.processFile()	Null Pointer Exception thrown	Null Pointer Exception thrown	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# FileHandler_DefaultConstructor
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests default constructor of FileHandler

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a FileHandler object with no arguments				
2	Assert that the FileHandler object isn’t Null	FileHandler defaultHandler	Not Null	Not Null	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# FileHandler_PreInputConstructor
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests constructor of FileHandler with a given file path name

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a FileHandler object with the given path name				
2	Assert that the FileHandler object isn't Null	FileHandler preInputHandler	Not Null	Not Null	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# FileHandler_OpenFile
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests opening files with a FileHandler object

Automated: yes_X_ no_X_
Results: Pass_X_ Fail__

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create File as control with desired path				
2	Open a file with FileHandler and the same path as in number 1				
3	Assert that the file pointers in steps 1 and 2 point to the same file	Fp, fp_in	"Project1/testing/csvTestFiles/testCPL.csv"	"Project1/testing/csvTestFiles/testCPL.csv"	
4	Open a file that will prompt the user for an input file with FileHandler				
5	Assert that the path returned in step 4 is correct	FileHandler defaultHandler	Project1/src/testCPL.csv	Project1/src/testCPL.csv	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# Voteable_getName
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests getName function in voteable class

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Voteable object created (IRCandidate) with name Bob and ID 101

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert voteable.getName is equal to Bob	voteable.getName()	“Bob”	“Bob”	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: **Unit_X_** **System__** **Test Date: 3/26/2023**
Test Case ID# Voteable_getNumVotes
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests getNumVotes function in voteable class

Automated: yes_X_ **no__**
Results: **Pass_X_** **Fail__**

Preconditions for Test:

Voteable object created (IRCandidate) with name Bob and ID 101

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert voteable.getNumVotes returns voteable.numVotes	voteable.getNumVotes()	0	0	

Postconditions for Test:

Project Name: Project 1: Voting System Team#16
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# Voteable_getID
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests getID function in voteable class

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Voteable object created (IRCandidate) with name Bob and ID 101

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert voteable.getID is equal to 101	voteable.getID()	101	101	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# Voteable_incrementVotes
Name(s) of Testers: Ethan Loukusa

Test Description:
Tests incrementVotes function in voteable class

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:

Voteable object created (IRCandidate) with name Bob and ID 101

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call voteable.incrementVotes(1)				
2	Assert voteable.getNumVotes equals the correct amount	voteable.getNumVotes()	1	1	
3	Call voteable.incrementVotes(10)				
4	Assert voteable.getNumVotes equals the correct amount	voteable.getNumVotes()	11	11	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**

Test Stage: Unit__ System_X_ Test Date: 3/26/2023

Test Case ID# IRElectionTest_NoBallotInputTest

Name(s) of Testers: Ethan Loukusa

Test Description:

Tests to make sure an IR election with no ballots in the csv file runs as expected. This means several things:

1. For IR elections, an election can run successfully with no ballots
2. Breaking ties works and is fair for IRElections

Automated: yes_X_ no__

Results: Pass_X_ Fail__

Preconditions for Test:

testIRNoBallots.csv is in the source file.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Read in testIRNoBallots.csv and create				

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit_X_ System__ Test Date: 3/26/2023
Test Case ID# IR_Produce_Audit_File
Name(s) of Testers: Gideon Tan

Test Description:
Tests to make sure the audit file is produced correctly after running an IR election. This is done manually for efficiency and simplicity.

Automated: yes__ no_X_
Results: Pass_X_ Fail__

Preconditions for Test:
Valid input files are in the directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program with corresponding test file as the input file	testIR.csv, testIR2.csv, testIR3.csv			
2	Check audit file production and compare it to manual calculation of election results		Manual calculation matches audit file	Manual calculation matches audit file	

Postconditions for Test:
Correct audit file is in the working directory

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X Test Date: 3/26/2023
Test Case ID# IR_getNumCandidates()
Name(s) of Testers: Robert Wang

Test Description:
Tests to make sure that the number of candidates is counted correctly when reading a file into an IRElection

Automated: yes_X no__
Results: Pass_X Fail__

Preconditions for Test:
Valid input files are in the directory and the respective IRElections are created for these files

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program with corresponding test file as the input file	testIRMain.csv, testIRTiedCandidates.csv, testIRNoMajorityWinner.csv, testIROneCandidate.csv, testIROneBallot.csv	4, 4, 4, 1, 2,	4, 4, 4, 1, 2,	
2	Check the output of the getNumCandidates() function for each election call, and see if it matches the actual number		Automatic calculation matches the numbers of candidates	Automatic calculation matches the number of candidates	

Postconditions for Test:

Correct IR file is in the working directory, and election information is not altered

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X Test Date: 3/26/2023
Test Case ID# IR_getNumBallots()
Name(s) of Testers: Robert Wang

Test Description:
Tests to make sure that the number of ballots is counted correctly in each of the IR files that are passed in as elections.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
Valid input files are in the directory and the respective IRElections are created for these files

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run program with corresponding test file as the input file	testIRMain.csv, testIROnecandidate.csv, testIROneBallot.csv testIRNoMajorityWinner.csv	6, 5, 1, 10	6, 5, 1, 10	
2	Check the function output of getNumBallots and see if it outputs the expected amount of ballots listed inside the IR files		Automatic comparison matches the input file ballot number	Automatic comparison matches the input file ballot number	

Postconditions for Test:
Correct IR file is in the working directory, and the file contents are not changed

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X Test Date: 3/26/2023
Test Case ID# IR_getCandidates()
Name(s) of Testers: Robert Wang

Test Description:
Tests to make sure the correct candidates are returned when getting the candidates from the election.

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
Valid input files are in the directory and the respective IRElections are created for these files

---

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program with corresponding test file as the input file	testIRMain.csv, testIROneCandidate.csv, testIRTiedCandidates.csv	{Rosen, Kleinberg, Chou, Royce}, {Rosen}, {Rosen, Royce}	{Rosen, Kleinberg, Chou, Royce}, {Rosen}, {Rosen, Royce}	
2	Create comparison IRCandidate[] arrays and loop through each index and check for equality with the returned values	{Rosen, Kleinberg, Chou, Royce}, {Rosen}, {Rosen, Royce}	Automatic comparison confirms equality between this input data and the correct IRCandidates	Automatic comparison confirms equality between this input data and the correct IRCandidates	

Project Name: Project 1: Voting System Team#16
Test Stage: Unit__ System_X Test Date: 3/26/2023
Test Case ID# IR_eliminateCandidates
Name(s) of Testers: Robert Wang

Test Description:
Tests to make sure that the correct candidates are eliminated when calling eliminateCandidates

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
Valid input files are in the directory and the respective IRElections are created for these files

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run program with corresponding test file as the input file	testIRMain.csv, testIROneCandidate.csv			
2	Calls the eliminate candidate function on each of the input IRElections		{Rosen, Kleinberg, Chou, Royce}, IllegalCallerException	{Rosen, Kleinberg, Chou, Royce}, IllegalCallerException	
3	Check for equality by looping through array	{Rosen, Kleinberg, Chou, Royce}, IllegalCallerException	Automatic Compares returns equal	Automatic compares returns equal	

Postconditions for Test:
Correct IR file still in directory and the numVotes for the last place candidate becomes -1, which indicates they are no longer being considered.

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X Test Date: 3/26/2023
Test Case ID# IR_runElection
Name(s) of Testers: Robert Wang

Test Description:
Test to make sure that the winner of the IRElections are calculated correctly

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test:
Valid input files are in the directory and the respective IRElections are created for these files

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run program with corresponding test file as the input file	testIRMain.csv, testIROneCandidate.csv, testIROneBallot.csv, testIRTiedCandidates.csv, testIRAlltied.csv	Rosen, Rosen, Rosen, (mix of Rosen, Royce), (mix of all candidates)	Rosen, Rosen, Rosen, (mix of Rosen, Royce), (mix of all candidates)	
2	For the IRElections with tied votes, test to see if the ties are broken somewhat equally	testIRTiedCandidates.csv, testIRAlltied.csv	(~500 : ~500), (~250, ~250, ~250, ~250)	(~500 : ~500), (~250, ~250, ~250, ~250)	
3	Check for equality by confirming if each candidate win count over 1000 runs of same election if within appropriate range		Automatic Compares returns within range	Automatic compares returns within range	

Postconditions for Test:
The Winner of the election is put to the front of the ranked candidates list, and the election run loop is terminated.

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X_ Test Date: 3/26/2023
Test Case ID# CPLElection_Normal
Name(s) of Testers: Gideon Tan

Test Description:
Ensures that runElection runs properly with a normal election

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test: FileHandler for testCPL.csv is created

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create CPLElection normal_election object from processFile	testCPL.csv			
2	Call runElection on normal_election	normal_election			
3	Assert that normal_election.results matches expected results		True	True	

Postconditions for Test:

Project Name: Project 1: Voting System **Team#16**
Test Stage: Unit__ System_X_ Test Date: 3/26/2023
Test Case ID# CPLElection_Two_tie
Name(s) of Testers: Gideon Tan

Test Description:
Ensures that runElection runs properly with an election with a tie between two parties

---

Automated: yes_X_ no__
Results: Pass_X_ Fail__

Preconditions for Test: FileHandler for testCPL.csv is created

---

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create CPLElection two_tie object from processFile	testCPLTwoTie.csv			
2	Call runElection on two_tie	two_tie			
3	Assert that two_tie .results matches expected results		True	True	

Postconditions for Test: