Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System __ **Test Date:**

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

Test Description:

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

Automated: yes___ no ___

Results: Pass Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2					
3					
4					

Post condition(s) for Test:

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#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_majority_candidate_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if majorityCandidate() within the IR class returns the proper candidate who has the majority.
Methods: majorityCandidate(), parseHeader(), processFile(), IR(FileReader, FileWriter)
Situation: Two candidates where one has the majority.
Input File: IRMajority1.csv **Test located in IRtests.java**

Automated: yes X no

Results: Pass X Fail_____

Preconditions for Test: The header, ballots have been processed. An IR instance has been created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRMajority1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRMajority1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRMajority1.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call majorityCandidate.		The Candidate A object should be returned.	The Candidate A object is returned.	

Post condition(s) for Test:
There are no changes to the system as a result of majorityCandidate() being called.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_majority_candidate_2 **Name(s) of Testers: Noreen Si**

Test Description: Tests if majorityCandidate() within the IR class returns null when there is an exact 50-50 split.
Methods: majorityCandidate(), parseHeader(), processFile(), IR(FileReader, FileWriter)
Situation: Two candidates where neither has a majority (both 50-50).
Input File: IRMajority2.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: The header, ballots have been processed. An IR instance has been created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRMajority2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRMajority2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRMajority2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call majorityCandidate.		null should be returned.	null is returned.	

Post condition(s) for Test:
There are no changes to the system as a result of majorityCandidate() being called.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_majority_candidate_3 **Name(s) of Testers: Noreen Si**

Test Description: Tests if majorityCandidate() within the IR class returns null when there are more than 2 candidates and none have the majority.
Methods: majorityCandidate(), parseHeader(), processFile(), IR(FileReader, FileWriter)
Situation: Three candidates where none have a majority.
Input File: IRMajority3.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail_____

Preconditions for Test: The header, ballots have been processed. An IR instance has been created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRMajority3.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRMajority3.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRMajority3.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	

Post condition(s) for Test:
There are no changes to the system as a result of majorityCandidate() being called.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_parseHeader_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if parseHeader() correctly initializes the curNumCandidates field.
Methods: parseHeader(), IR(FileReader, FileWriter)
Situation: The curNumCandidates field must be initialized to 4.
Input File: IRHeader1.csv **Test is located in IRtests.java**

Automated: yes X no __

Results: Pass X Fail

Preconditions for Test: The input file has been opened. The header has been parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRHeader1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRHeader1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Check if the IR instance's curNumCandidates field equals 4 through an assertion.		curNumCandidates = 4	curNumCandidates = 4	

Post condition(s) for Test:
The IR fields relating to the header, such as curNumCandidates, numBallots, etc. have been initialized properly.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_parseHeader_2 **Name(s) of Testers: Noreen Si**

Test Description: Tests if parseHeader() correctly initializes the numBallots field.

Methods: parseHeader(), IR(FileReader, FileWriter)

Situation: The numBallots field must be initialized to 6.

Input File: IRHeader1.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail ___

Preconditions for Test: The input file has been opened. The header has been parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRHeader1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRHeader1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Check if the IR instance's numBallots field has been stored as 6.		numBallots = 6	numBallots = 6	

Post condition(s) for Test:

The IR fields relating to the header, such as curNumCandidates, numBallots, etc. have been initialized properly.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_parseHeader_3 **Name(s) of Testers: Noreen Si**

Test Description: Tests if parseHeader() correctly initializes the Candidates.
Methods: parseHeader(), IR(FileReader, FileWriter)
Situation: parseHeader() must properly read header information relating to the candidates.
Input File: IRHeader1.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail _____

Preconditions for Test: The input file has been opened. The header has been parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRHeader1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRHeader1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Check if the Candidate object in index 2 has the name “Chou”		name of candidate with index 2: Chou	name of candidate with index 2: Chou	

Post condition(s) for Test:
The IR fields relating to the header, such as curNumCandidates, numBallots, etc. have been initialized properly.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_reassignVotesNoneEliminated_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if reassignVotes properly reassigns ballots to another candidate.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter), reassignVotes()
Situation: There are two candidates and none have been eliminated. Candidate A's ballots are transferred to Candidate B's Tree, if applicable.
Input File: IRReassignNoneEliminated1.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRReassignNoneEliminated1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRReassignNoneEliminated1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with FileReader, BufferedReader objects.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IIRReassignNoneEliminated1.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call reassignVotesNoneEliminated() to transfer votes.		There is no error when calling reassignVotesNoneEliminated()	There is no error when calling reassignVotesNoneEliminated()	
7	Check if candidate B has 4 votes.		Candidate B has 4 votes.	Candidate B has 4 votes.	

Post condition(s) for Test:
Candidate B has gained new ballots in its Tree, and the vote count is incremented for B.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_reassignVotes_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if reassignVotes properly reassigns ballots to another candidate in multiple rounds (one candidate previously reassigns votes to the other candidates).
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter), reassignVotes(), reassignVotesNoneEliminated()
Situation: There are three candidates and Candidate A is first eliminated, then B.
Input File: IRReassign1.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail ____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRReassign1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRReassign1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRReassign1.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call reassignVotes() for that IR instance for A to the B and C Trees.	BNode_wave1, CNode_wave1	There are no errors when calling reassignVotes()	There are no errors when calling reassignVotes()	
7	Call reassignVotes() for that IR instance for B to the C Tree.	ANode_wave2, CNode_wave2	There are no errors when calling reassignVotes()	There are no errors when calling reassignVotes()	
8	Check if Candidate C has 10 votes.		Candidate C now has 10 votes.	Candidate C now has 10 votes.	

Post condition(s) for Test:

Candidate B, C have gained new ballots in their Trees, and the vote count is incremented for them.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_eliminate_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if eliminateCandidate() properly conducts reassignment.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter), eliminateCandidate(int)
Situation: There are two candidates and none have been eliminated. Candidate A's ballots are transferred to Candidate B's Tree, if applicable, as in test_reassignVotes_1.
Input File: IREliminate1.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IREliminate1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IREliminate1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IREliminate1.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call eliminateCandidate for A.		There are no errors when calling eliminateCandidate()	There are no errors when calling eliminateCandidate()	
7	Call eliminateCandidate for B.		There are no errors when calling eliminateCandidate()	There are no errors when calling eliminateCandidate()	
8	Check if C has 10 votes.		Candidate C has 10 votes.	Candidate C has 10 votes.	

Post condition(s) for Test:
Candidate B, C have gained new ballots in their Trees, and the vote count is incremented for them.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_reassignVotes_2 **Name(s) of Testers: Noreen Si**

Test Description: Tests if reassignVotes properly reassigns ballots to another candidate when there are multiple eliminated candidates in the ballot.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter), reassignVotes()
Situation: Transfer Candidate A's votes to Candidate B when candidates at indices 2, 4 were previously eliminated.
Input File: IRReassign2.csv **Test is located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail ___

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRReassign2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRReassign2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRReassign2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Set candidates at indices 2, 4 as eliminated. (eliminated = true)		There are no errors when setting candidates as eliminated.	There are no errors when setting candidates as eliminated.	
7	Reassign votes from Candidate A to Candidate B.	ballotNode, toInsert, eliminated	There are no errors when reassignVotes().	There are no errors when reassignVotes().	
8	See if Candidate B now has 2 votes.		Candidate B has 2 votes.	Candidate B has 2 votes.	

Post condition(s) for Test:
Candidate B has gained new ballots in its Tree, and the vote count is incremented for it.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_eliminate_2 **Name(s) of Testers: Noreen Si**

Test Description: Tests if eliminateCandidate() properly conducts reassignment.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter), eliminateCandidate(int)
Situation: Transfer Candidate A's votes to Candidate B when candidates at indices 2, 4 were previously eliminated, as in test_reassignVotes_2.
Input File: IREliminate2.csv **Test located in IRtests.java**

Automated: yes X no ___

Results: Pass X Fail ___

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IREliminate2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IREliminate2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IREliminate2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Set candidates at indices 2, 4 as eliminated.		There are no errors when setting candidates as eliminated.	There are no errors when setting candidates as eliminated.	
7	Call eliminateCandidate() fo candidate A.		There are no errors when calling eliminateCandidate().	There are no errors when calling eliminateCandidate().	
8	See if Candidate B now has 2 votes.		Candidate B has 2 votes.	Candidate B has 2 votes.	

Post condition(s) for Test:
Candidate B has gained new ballots in its Tree, and the vote count is incremented for it.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_1 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with a simple majority.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()
Input File: IRConductAlgorithm1.csv
Output File: Audit_IRConductAlgorithm1.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail _____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRConductAlgorithm1.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRConductAlgorithm1.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRConductAlgorithm2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate A printed to screen with 2 votes	Candidate A printed to screen with 2 votes	

Post condition(s) for Test:
The election has concluded with Candidate A being announced as the winner after all necessary reassignments and/or eliminations.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_2 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with no immediate majority nor ties, and must eliminate a candidate to decide the winner.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()
Input File: IRConductAlgorithm2.csv
Output File: Audit_IRConductAlgorithm2.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRConductAlgorithm2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRConductAlgorithm2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRConductAlgorithm2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate B printed to screen with 6 votes.	Candidate B printed to screen with 6 votes.	

Post condition(s) for Test:
The election has concluded with Candidate B being announced as the winner after all necessary reassignments and/or eliminations.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_3 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with no immediate majority with one tie for elimination

Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()

Input File: IRConductAlgorithm3.csv

Output File: Audit_IRConductAlgorithm3.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRConductAlgorithm2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRConductAlgorithm2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRConductAlgorithm2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate B printed to screen with 5 votes.	Candidate B printed to screen with 5 votes.	This is ONLY the expected result during development, as breakTie is required, which delivers randomized results.

Post condition(s) for Test:

The election has concluded with Candidate B being announced as the winner after all necessary reassignments and/or eliminations.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_4 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with no immediate majority with multiple rounds of numerous tied candidates.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()
Input File: IRConductAlgorithm4.csv
Output File: Audit_IRConductAlgorithm4.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRConductAlgorithm2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRConductAlgorithm2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRConductAlgorithm2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate A wins with 11 votes printed to screen.	Candidate A printed to screen with 11 votes.	This is ONLY the expected result during development, as breakTie is required, which delivers randomized results.

Post condition(s) for Test:
The election has concluded with Candidate A being announced as the winner after all necessary reassignments and/or eliminations.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_5 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with popularity deciding the winner.
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()
Input File: IRConductAlgorithm5.csv
Output File: Audit_IRConductAlgorithm5.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail _____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	IRConductAlgorithm2.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	IRConductAlgorithm2.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	IRConductAlgorithm2.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate A wins with 3 votes (printed to the screen).	Candidate A printed to screen with 3 votes.	

Post condition(s) for Test:
The election has concluded with Candidate A being announced as the winner.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/22/2023**

Test Case ID#: test_conductAlgorithm_6 **Name(s) of Testers: Noreen Si**

Test Description: Tests if conductAlgorithm() can successfully conduct an IR election with a large number of ballots (1000 ballots).
Methods: parseHeader(), processFile(), IR(FileReader, FileWriter, BufferedReader), conductAlgorithm()
Input File: ballots.csv
Output File: Audit_IRConductAlgorithm6.txt **Test located in IRtests.java**

Automated: yes no X

Results: Pass X Fail_____

Preconditions for Test: The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	ballots.csv	A FileReader has successfully been created.	Successful creation of the FileReader.	
2	Create a BufferedReader and read one line to advance file pointer.	ballots.csv (Created with FileReader from previous step)	A BufferedReader has been successfully created.	A BufferedReader has been successfully created.	
3	Create an instance of IR with this FileReader object.		An IR instance has successfully been instantiated.	The IR instance was created.	
4	Parse the header for this IR instance.		The header has successfully been parsed for the two candidates.	The header was successfully read.	
5	Process the ballots.	ballots.csv	The ballots have been successfully processed.	The ballots have been successfully processed.	
6	Call conductAlgorithm()		Candidate G wins with 373 votes printed to screen.	Candidate G printed to screen with 373 votes.	

Post condition(s) for Test:
The election has concluded with Candidate G being announced as the winner after all necessary reassignments and/or eliminations.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: getName() **Name(s) of Testers: Noreen Si**

Test Description: Tests if getName() correctly returns the candidate's name.

Methods: getName()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate Object.		Object successfully initialized	Object successfully initialized	
2	Check if the name is "CandidateC"		Name returned by getName() is equal to "CandidateC"	Name returned by getName() is equal to "CandidateC"	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: getParty() **Name(s) of Testers: Noreen Si**

Test Description: Tests if getParty() correctly returns the candidate's party.

Methods: getParty()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate Object.		Object successfully initialized	Object successfully initialized	
2	Check if the party is "(C)"		Party name returned by getParty is "(C)"	Party name returned by getParty is "(C)"	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: getNumVotes() **Name(s) of Testers: Noreen Si**

Test Description: Tests if getNumVotes() correctly returns the candidate's number of votes.

Methods: getVotes()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate Object.		Object successfully initialized	Object successfully initialized	
2	Check if the number of votes is 1		Number is one	Number is one	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: isEliminated() **Name(s) of Testers: Noreen Si**

Test Description: Tests if isEliminated() successfully returns whether or not the candidate has been eliminated.

Methods: isEliminated()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate Object.		Object successfully initialized	Object successfully initialized	
2	Check if Candidate isEliminated()		False	False	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: setEliminated() **Name(s) of Testers: Noreen Si**

Test Description: Tests if setEliminated() successfully returns whether or not the candidate has been eliminated.

Methods: setEliminated()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate Object.		Object successfully initialized	Object successfully initialized	
2	Use setEliminated(true) on Candidate object		Candidate successfully set as eliminated	Candidate successfully set as eliminated	
3	Check if Candidate isEliminated()		True	True	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run and the Candidate's eliminated field being set to true.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 3/25/2023**

Test Case ID#: isEliminated() **Name(s) of Testers: Noreen Si**

Test Description: Tests if isEliminated() successfully returns whether or not the candidate has been eliminated.

Methods: getVotes()

No extra files used. **Test located in CandidateTest.java**

Automated: yes X no

Results: Pass X Fail __

Preconditions for Test: The Candidate object has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize the Tree for the Candidate.		Tree successfully initialized.	Tree successfully initialized	
2	Create a Candidate Object with the Tree.		Object successfully initialized	Object successfully initialized	
3	Check if Candidate getBallots() is the same Tree as the one initialized.		Tree matches.	Tree matches.	

Post condition(s) for Test:

None, other than the Candidate object having been initialized prior to this test's run.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit X System __ **Test Date: 03/26/2023**

Test Case ID#: test_parseHeader() **Name(s) of Testers: Lane Enget**

Test Description: Tests that the parseHeader function sets the parties, total seats, and vote totals in the system.
Methods: parseHeader(), CPL(reader1, null, br1)

Test located in CPLTests.java

Automated: yes_x_ no ___

Results: Pass X Fail

Preconditions for Test:
The input file has been opened. The header has been parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	basic.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	basic.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Check party names have been correctly initialized		Parties are “PartyA.” “PartyB,” and “PartyC”	Parties are “PartyA.” “PartyB,” and “PartyC”	
5	Check total available seat number has been correctly initialized		Total seats equals 1	Total seats equals 1	
6	Check vote total has been correctly initialized		Vote total equals 6	Vote total equals 6	
7	Check party candidates have been correctly initialized		Candidates are “Bob,” “Mike,” and “Sam”	Candidates are “Bob,” “Mike,” and “Same”	

Post condition(s) for Test:

File is ready to be processed for ballot information

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _X_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_processFile() **Name(s) of Testers: Lane Enget**

Test Description: Tests that the processFile function correctly allocates votes to the appropriate parties.
Methods: parseHeader(), processFile(), CPL(reader2, null, br2)

Test located in CPLTests.java

Automated: yes X no

Results: Pass __X___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	basic.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	basic.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Check party votes have been correctly initialized		Party votes are 1, 4, and 1	Party votes are 1, 4, and 1	

Post condition(s) for Test:

CPL algorithm is ready to be conducted on the ballot data

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _X_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_processFile_2() **Name(s) of Testers: Lane Enget**

Test Description: Tests that the conductAlgorithm() function correctly allocates the appropriate seats to the right parties. Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader3, null, br3)

Test located in CPLTests.java

Automated: yes__x__ no ___

Results: Pass X Fail

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	basic.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	basic.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check each party's received seats		Seats received equals 0, 1, and 0	Seats received equals 0, 1, and 0	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _X_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_initializeParty() **Name(s) of Testers: Lane Enget**
Test Description: Tests that initializeParty() creates Party objects after the parties are read from the input file.
Methods: parseHeader(), initializeParty(String[] parties), CPL(reader4, null, br4)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass __X___ Fail _____

Preconditions for Test:
The input file has been opened. The header has been parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	basic.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	basic.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call initializeParty() with the party array parameter		initializeParty() creates PartyD, PartyE, and PartyF	initializeParty() creates PartyD, PartyE, and PartyF	
6	Check party objects are correctly named		getName() equals PartyD, PartyE, and PartyF	getName() equals PartyD, PartyE, and PartyF	

Post condition(s) for Test:

CPL algorithm is ready to be conducted on the ballot data and every Party object is correctly instantiated.

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _X_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_drawLotto() **Name(s) of Testers: Lane Enget**

Test Description: Tests that drawLotto() randomly assigns seats in the event where a party has earned more seats than candidates.
Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader5, null, br5)

Test located in CPLTests.java

Automated: yes x no

Results: Pass __X___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	lottoCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	lottoCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check each party's received seats		Seats received are 1 and 1	Seats received are 1 and 1	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_break_two_tie() **Name(s) of Testers: Lane Enget**

Test Description: Tests that breakTie() works in the event there is a 2-way tie between parties

Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader6, null, br6)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass X Fail

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	twoTieCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	twoTieCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that the seats were distributed and the tie was broken		Seats received are 1 for the 0th party and 1 for either the 1st or 2nd party	Seats received are 1 for the 0th party and 1 for either the 1st or 2nd party	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_break_multi_tie() **Name(s) of Testers: Lane Enget**

Test Description: Tests that breakTie() works in the event there is a tie between multiple parties

Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader7, null, br7)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test:

The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	multiTieCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	multiTieCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that the seats were distributed and the tie was broken		Seats received are 1 for the 0th party and 1 for the 1st party, 2nd party, or 3rd party	Seats received are 1 for the 0th party and 1 for the 1st party, 2nd party, or 3rd party	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_break_lotto_tie() **Name(s) of Testers: Lane Enget**

Test Description: Tests the case where there are both ties and lotteries in an election.
Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader8, null, br8)

Test located in CPLTests.java

Automated: yes__x__ no ___

Results: Pass ___x__ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	lottoTieCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	lottoTieCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that seats were distributed and that ties and lotteries were handled correctly		Seats received for the 0th party are 1 and seats received for the 1st and 2nd party are either 2 and 0, 1 and 1, or 0 and 2 respectively	Seats received for the 0th party are 1 and seats received for the 1st and 2nd party are either 2 and 0, 1 and 1, or 0 and 2 respectively	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_full() **Name(s) of Testers: Lane Enget**

Test Description: Tests the case where there are exactly the same number of seats as total candidates (everyone wins)
Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader9, null, br9)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	fullCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	fullCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that distributed seats match the exact amount of candidates in the election		Seats received for all parties is 1	Seats received for all parties is 1	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_knockout() **Name(s) of Testers: Lane Enget**

Test Description: Tests the case of a landslide victory – where every ballot goes to the same party
Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader10, null, br10)

Test located in CPLTests.java

Automated: yes__x_ no ___

Results: Pass__x___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	knockoutCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	knockoutCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that every seat is assigned to one party		Seats received for the 0th party are 2 and seats received for both the 1st and 2nd party are 0	Seats received for the 0th party are 2 and seats received for both the 1st and 2nd party are 0	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit _x_ System __ **Test Date: 03/26/2023**

Test Case ID#: test_all_tie() **Name(s) of Testers: Lane Enget**

Test Description: Tests the case where every party receives the same amount of votes

Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader11, null, br11)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail _____

Preconditions for Test:

The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	allTieCase.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	allTieCase.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that seats were distributed randomly in the event all parties receive the same amount of votes		Seats received for each party are greater than or equal to 1 and the total amount of seats allocated is 4	Seats received for each party are greater than or equal to 1 and the total amount of seats allocated is 4	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System _x_ **Test Date: 03/26/2023**

Test Case ID#: test_basic_file() **Name(s) of Testers: Lane Enget**

Test Description: Tests the whole CPL election process against a basic input file.
Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader12, null, br12)

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	basic.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	basic.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that the parties have been initialized properly, the seats have been completely distributed, and the vote total is correct		Parties are called “PartyA,” “PartyB,” and “PartyC.” Total seats left to be distributed is 0. Total amount of votes received is 6. PartyA’s candidate is Bob. PartyB received 1 seat.	Parties are called “PartyA,” “PartyB,” and “PartyC.” Total seats left to be distributed is 0. Total amount of votes received is 6. PartyA’s candidate is Bob. PartyB received 1 seat.	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System _X_ **Test Date: 03/26/2023**

Test Case ID#: test_france() **Name(s) of Testers: Lane Enget**

Test Description: Tests the CPL election process against a French input file

Methods: parseHeader(), processFile(), conductAlgorithm(), CPL(reader13, null, br13)

Test located in CPLTests.java

Automated: yes_X__ no_____

Results: Pass __X___ Fail_____

Preconditions for Test:
The input file has been opened. The header has been parsed. The file has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a FileReader object	france.csv	A file reader has been successfully created	A file reader has been successfully created	
2	Create a BufferedReader object	france.csv	A buffered reader has been successfully created	A buffered reader has been successfully created	
3	Create a CPL object with the FileReader and BufferedReader objects		A CPL object has been successfully created	A CPL object has been successfully created	
4	Call parseHeader()		parseHeader() is successfully called	parseHeader() is successfully called	
5	Call processFile()		processFile() is successfully called	processFile() is successfully called	
6	Call conductAlgorithm()		conductAlgorithm() is successfully called	conductAlgorithm() is successfully called	
7	Check that the parties have been initialized properly, the seats have been completely distributed, and the vote total is correct		Parties are called “La France Insoumise,” “Les Verts,” and “En Marche,” “Les Republicains,” and “Rassemblement National.” Total seats left to be distributed	Parties are called “La France Insoumise,” “Les Verts,” and “En Marche,” “Les Republicains,” and “Rassemblement National.” Total seats left to be distributed is 0. Total amount of votes received is 25. La France Insoumise has candidate Melenchon.	

			is 0. Total amount of votes received is 25. La France Insoumise has candidate Melenchon. En Marche has candidates Macron and Lagarde. En Marche receives at least 1 seat, La France Insoumise receives 1 seat, and Ressemblement National receives 1 seat.	En Marche has candidates Macron and Lagarde. En Marche receives at least 1 seat, La France Insoumise receives 1 seat, and Ressemblement National receives 1 seat.	

Post condition(s) for Test:

N/A

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System _x_ **Test Date: 03/26/2023**

Test Case ID#: test_incrementSeatCount() **Name(s) of Testers: Pyrenees Gavois**

Test Description: Tests the incrementSeatCount() function of a Party object

Methods: Party("PartyA")
setSeatsReceived()
incrementSeatCount()
getSeatsReceived()

Test located in CPLTests.java

Automated: yes x no

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Party object with name "PatyA"		A Party object has been created	A Party object has been created	
2	Set the party seats to 5		PartyA has 5 seats	PartyA has 5 seats	
3	Invoke the increment seats function		PartyA has 6 seats	PartyA has 6 seats	
4	Check to see if the party seats are now 6		PartyA has 6 seats and is asserted as true	PartyA has 6 seats and is asserted as true	

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System _x_ **Test Date:** 03/26/2023

Test Case ID#: test_incrementVoteCount() **Name(s) of Testers:** Pyrenees Gavois

Test Description: Tests the incrementVoteCount() function of a Party object

Methods: Party("PartyB")
setNumVotes()
incrementVoteCount()
getNumVotes()

Test located in CPLTests.java

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Party object with name "PatyB"		A Party object has been created	A Party object has been created	
2	Set the party vote count to 20		PartyB has 20 votes	PartyB has 20 votes	
3	Invoke the increment votes function		PartyB has 21 votes	PartyB has 21 votes	
4	Check to see if the party votes are now 21		PartyB has 21 votes and is asserted as true	PartyB has 21 votes and is asserted as true	

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System Yes

Test Date: 3/26/2023

Test Case ID#: CPL System test
Test Description: Testing CPL system functionality

Name(s) of Testers: Jonathan Haak

Tests are in MainTests.java, tests make use of RunElection class

Automated: yes___ no _X_

Results: Pass X Fail

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Create Run Election object with file name as parameter for constructor	france.csv (located in CPLTestsResources under Testing)			Please put france.csv in src/VotingSystem prior to testing
3	Call start on Run Election object		Election runs, audit file is created, and results are output	Election runs, audit file is created, and results are output	
4					

Post condition(s) for Test:

Results for election are present on terminal, and audit file containing election results is present in program directory

Project Name: Project 1: Voting System **Team#18**

Test Stage: Unit __ System Yes **Test Date: 3/26/2023**

Test Case ID#: IR System test **Name(s) of Testers: Jonathan Haak**

Test Description: Testing IR system functionality

Tests are in MainTests.java, tests make use of RunElection class

Automated: yes___ no _X_

Results: Pass X Fail

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Create Run Election object with file name as parameter for constructor	ballots.csv (located in IRTestsResources under Testing)			Please put ballots.csv in the src/VotingSystem prior to test
3	Call start on Run Election object		Election runs, audit file is created, and results are output	Election runs, audit file is created, and results are output	
4					

Post condition(s) for Test:

Results for election are present on terminal, and audit file containing election results is present in program directory