Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will test the getters and setters for currDis and id data members.

The tests are stored in the ballot_unittest.cc file.
The methods used are SetCurrDis, SetId, GetCurrDis, GetId

Automated: yes _✓__ no ___

Results: Pass __✓___ Fail ____

Preconditions for Test:

- **ballot.h and ballot.cpp must compile**
- **Parameters for SetCurrDis and SetId are integers**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetCurrDis method	none	0	0	
2	Testing the SetCurrDis method	1	1	1	
3	Testing the GetId method	none	1	1	
4	Testing the SetId method	99	99	99	

Post condition(s) for Test:

- id and currDis data members are set to the desired values
- Desired values are returned for GetId() and SetId()

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will test that the AddCandidate method properly adds a candidate string to the candidates vector

The test is stored in the ballot_unittest.cc file.
The methods used are AddCandidate and Print

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **ballot.h and ballot.cpp must compile**
- **Parameter for AddCandidate must be a string**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the AddCandidate method	"Trump"	"Trump\n"	"Trump\n"	

Post condition(s) for Test:

- A candidate is added to the candidates vector of the ballot class

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will test whether or not the Print method prints out the elements of the candidates vector in the correct order

The test is stored in the ballot_unittest.cc file.
The methods used are AddCandidate and Print

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test:

- **ballot.h and ballot.cpp must compile**
- **Parameter for AddCandidate must be a string**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the Print method	"Trump" "Biden" "Harris"	"Trump\nBiden\nHarris\n"	"Trump\nBiden\nHarris\n"	

Post condition(s) for Test:

- The candidates vector is populated with the correct candidates
- The elements of the candidates vector are printed out in the correct order

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the getters and setters for the Candidate class work properly.

The tests are stored in the candidate_unittests.cpp file. The methods used are GetName, GetParty, SetName, SetParty, GetBallotListSize

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **Parameters for SetName and SetParty must strings**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetName method	none	"Bui" "Nguyen"	"Bui" "Nguyen"	
2	Testing the GetParty method	none	"R" "D"	"R" "D"	
3	Testing the SetName method	"Tran"	"Tran"	"Tran"	
4	Testing the SetParty method	"D"	"D"	"D"	
5	Testing the GetBallotListSize method	none	0	0	

Post condition(s) for Test:

- A candidate object's name and party are set to desired values
- Desired values for party, name, ballot list size are returned

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the AddBallot method properly adds a ballot to the ballots vector for a candidate

The tests are stored in the candidate_unittests.cpp file.
The methods used are AddBallot and GetBallotListSize

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **Parameter must be a pointer to a ballot object**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Added a ballot to the ballots vector and returned the vector's size	newBallot	1	1	

Post condition(s) for Test:

- A ballot is added to a candidate object's ballots vector

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the RemoveBallot method properly removes a ballot to the ballots vector for a candidate

The tests are stored in the candidate_unittests.cpp file. The methods used are AddBallot, RemoveBallot and GetBallotListSize

Automated: yes__✓_ no ___

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **A ballot object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Removed a ballot from an empty ballots vector to if it returns null.	none	NULL	NULL	
2	Added a ballot to the ballots vector and returned the vector's size.	newBallot	1	1	
3	Removed a ballot from the ballot vector and checked if it removed the correct ballot	none	newBallot	newBallot	
3	Checked the ballot vector's size of removing a ballot	none	0	0	

Post condition(s) for Test:

- A ballot is removed from a candidate object's ballots vector

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the getter and setter for fileName work properly

The test is stored in the driver_unittests.cpp file.
The methods used are GetFileName and SetFileName

Automated: yes ✓ no

Results: Pass ___✓__ Fail_____

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetFileName method	none	“ballots”	“ballots”	
2	Testing the SetFileName method	“votes”	“votes”	“votes”	

Post condition(s) for Test:

- The correct file name is returned
- fileName is set to the right name

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

Test Case ID#: Driver_Invalid_Input
Test Description: This test will determine if driver class handles an invalid input correctly

Name(s) of Testers: Hoai Bui

The test is stored in the driver_unittests.cpp file. The methods used are ReadInElectionType(), ReadInCandidates(), ReadInBallots()

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must exist**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing to see what is outputted when the file name inputted is not a CSV file	"Invalid"	"File Handle is not open for Election Type.\nDid not recognize election type: NONE\nFIle not open.\n"	"File Handle is not open for Election Type.\nDid not recognize election type: NONE\nFIle not open.\n"	

Post condition(s) for Test:

- The correct error statements are outputted

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the read methods work properly for an instant runoff election.

The tests are stored in the driver_unittests.cpp file. The methods used are ReadInElectionType, ReadInNumCandidates, ReadInCandidates, ReadInNumberOfBallots, ReadInBallots

Automated: yes__✓_ no ___

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Driver class is constructed	ir.csv	none	none	
2	Testing ReadInElectionType	ir.csv	"Compute IR election.\n"	"Compute IR election.\n"	
3	Testing ReadInNumCandidates	ir.csv	"Number of candidates: 4\n"	"Number of candidates: 4\n"	
4	Testing ReadInCandidates	ir.csv	0	0	
5	Testing ReadInNumberOfBallots	ir.csv	"Number of ballots: 9\n"	"Number of ballots: 9\n"	
6	Testing ReadInBallots	ir.csv	0	0	

Post condition(s) for Test:

- All data is read from the CSV file for an instant runoff election.

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the read methods work properly for an OPL election.

The tests are stored in the driver_unittests.cc file. The methods used are ReadInElectionType, ReadInNumCandidates, ReadInCandidates, ReadInNumberOfBallots, ReadInBallots, ReadInNumberOfSeats

Automated: yes__✓_ no ___

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Driver class is constructed	opl.csv	none	none	
2	Testing ReadInElectionType	opl.csv	"Compute OPL election.\n"	"Compute OPL election.\n"	
3	Testing ReadInNumCandidates	opl.csv	"Number of candidates: 6\n"	"Number of candidates: 6\n"	
4	Testing ReadInCandidates	opl.csv	0	0	
5	Testing for ReadInNumberOfSeats	opl.csv	“Number of seats: 3\n”	“Number of seats: 3\n”	
6	Testing ReadInNumberOfBallots	opl.csv	"Number of ballots: 9\n"	"Number of ballots: 9\n"	
7	Testing ReadInBallots	opl.csv	0	0	

Post condition(s) for Test:

- All data is read from the CSV file for an instant OPL election.

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-25-2021**

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

Test Description: This test will determine if the ProcessCSV method works properly.

The test is stored in the driver_unittests.cpp file.
The method used is ProcessCSV.

Automated: yes ✓ no

Results: Pass ___✓__ Fail_____

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	ProcessCSV is run on a CSV file for an IR election	ir.csv	"Compute IR election.\nNumber of candidates: 4\nNumber of ballots: 9\n"	"Compute IR election.\nNumber of candidates: 4\nNumber of ballots: 9\n"	
2	ProcessCSV is run on a CSV file for an OPL election	opl.csv	"Compute OPL election.\nNumber of candidates: 6\nNumber of seats: 3\nNumber of ballots: 9\n"	"Compute OPL election.\nNumber of candidates: 6\nNumber of seats: 3\nNumber of ballots: 9\n"	
3	ProcessCSV is run with an invalid file name	“ballots”	"File Handle is not open for Election Type.\nDid not recognize election type: NONE\nElection type not recognized.\nFIle not open.\n"	"File Handle is not open for Election Type.\nDid not recognize election type: NONE\nElection type not recognized.\nFIle not open.\n"	

Post condition(s) for Test:

- CSV file is properly processed and proper statements are outputted
- Invalid input file name is handled properly and proper statements are outputted

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-25-2021**

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

Test Description: This test will determine if the ParseLine method works properly, populating a specified vector with values from a comma-separated in a line of text.

The test is stored in the driver_unittests.cpp file.
The method used is ParseLine.

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the ParseLine method with a line of text	"one, two, three, four"	“one” “two” “three” “four”	“one” “two” “three” “four”	

Post condition(s) for Test:

- Values from a line of text are properly added to the specified vector in the correct order

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-25-2021**

Test Case ID#: Driver_ParseLine2 **Name(s) of Testers: Hoai Bui**

Test Description: This test will determine if the ParseLine2 method works properly, populating a specified vector with values from a comma-separated in a line of text.

The test is stored in the driver_unittests.cpp file.
The method used is ParseLine2.

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the ParseLine method with a line of text	"four, three, two, one"	"four" "three" "two" "one"	"four" "three" "two" "one"	

Post condition(s) for Test:

- Values from a line of text are properly added to the specified vector in the correct order

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit _✓_ System __ **Test Date: 3-13-2021**

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

Test Description: This test will determine if the GetOPLVote method works properly, returning the index of the string in which a 1 appears.

The test is stored in the driver_unittests.cc file.
The method used is GetOPLVote.

Automated: yes _✓_ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **candidate.h and candidate.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **driver.h and driver.cpp must compile**
- **election.h and election.cpp must compile**
- **A driver object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetOPLVote method with string containing commas and a 1	“,,1”	2	2	
2	Testing the GetOPLVote method with string containing only a 1 and no commas	“1”	0	0	
3	Testing the GetOPLVote method with string containing a comma and no 1’s	“,”	-1	-1	
4	Testing the GetOPLVote method with an empty string	“”	-1	-1	

Post condition(s) for Test:

- The index in which a 1 appears in a string is returned
- If there is no 1 in the string, -1 is returned as the index

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Getter_Setters
Test Description: This test will test all the getter and setters numberOfCandidates, numberOfBallots, numberOfSeats, and quota.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file. The methods used are GetNumberOfSeats, SetNumberOfSeats, GetNumberOfBallots, SetNumberOfBallots, GetNumberOfCandidates, SetNumberOfCandidates, GetQuota, SetQuota

Automated: yes_✓__ no ___

Results: Pass _✓_____ Fail _____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **Parameters for numberOfCandidates, numberOfBallots, numberOfSeats, and quota are integers**
- **An election object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetNumberOfSeats method	none	3	3	
2	Testing the GetNumberOfSeats method	none	-1	-1	
3	Testing the SetNumberOfSeats method	3	3	3	
4	Testing the GetNumberOfBallots method	none	9	9	
5	Testing the GetNumberOfBallots method	none	-1	-1	
6	Testing the SetNumberOfBallots method	6	6	6	

7	Testing the GetNumberOfCandidates method	none	6	6	
8	Testing the GetNumberOfCandidates method	none	-1	-1	
9	Testing the SetNumberOfCandidates method	6	6	6	
10	Testing the GetQuota method	none	3	3	
11	Testing the GetQuota method	none	-1	-1	
12	Testing the SetQuota method	3	3	3	

Post condition(s) for Test:

- numberOfCandidates, numberOfBallots, numberOfSeats, and quota data members are set to the desired values
- Getters for numberOfCandidates, numberOfBallots, numberOfSeats, and quota data members return correct values

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

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

Test Description: This test will test the getter and setter of the electionType.

The tests are stored in the election_unittest.cpp file.
The methods used are GetElectionType and SetElectionType

Automated: yes_✓__ no ___

Results: Pass ___✓__ Fail _____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **parameters for SetElectionType are a string**
- **An election object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetElectionType method	none	"NONE"	"NONE"	
2	Testing the GetElectionType method	"OPL"	"OPL"	"OPL"	
3	Testing the SetElectionType method	"OPL"	"OPL"	"OPL"	

Post condition(s) for Test:

- The electionType data member is set to the correct value and the getter returns the correct election type

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Get_Set_VotesForParty
Test Description: This test will test the getter and setter votes for parties.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file.
The methods used are GetVotesForParty, SetVotesForParties, AddCandidate, AddBallot, GetName, SetNumberOfCandidates, IncreaseNumberOfCandidates, AddParty

Automated: yes ✓ no __

Results: Pass ✓ Fail

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **Parameters for GetVotesForParty are strings**
- **An election object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetVotesForParty method	'D'	1	1	
2	Testing the SetVotesForParties method	none	1	1	

Post condition(s) for Test:

- The votes for each party are returned/set

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Increaser
Test Description: This test will test the IncreaseNumberOfCandidates and IncreaseNumberOfBallots methods to determine if they correctly increment the numberOfBallots and numberOfCandidates data members by 1.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file. The methods used are IncreaseNumberOfCandidates, IncreaseNumberOfBallots, SetNumberOfBallots, SetNumberOfCandidates, GetNumberOfCandidates, GetNumberOfBallots

Automated: yes ✓ no

Results: Pass _✓____ Fail_____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **numberOfCandidates and numberOfBallots are integers**
- **An election object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the IncreaseNumberOfCandidates method	6	7	7	
2	Testing the IncreaseNumberOFBallots method	9	10	10	

Post condition(s) for Test:

- numberOfCandidates and numberOfBallots are incremented by 1

Project Name: Project 1: Voting System **Team# 11**

Test Stage: Unit _✓_ System __

Test Date: 3-13-2021

Test Case ID#: Election_Add_Remove_Candidates
Test Description: This test will determine the RemoveCandidate and AddCandidate methods work properly, doing what their names specify.

Name(s) of Testers: Emma Spindler

Automated: yes_✓__ no ___

The tests are stored in the election_unittest.cpp file. The methods used are AddCandidate, RemoveCandidate, GetNumberOfCandidates, SetNumberOfCandidates, IncreaseNumberOfCandidates

Results: Pass _✓_____ Fail _____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **An election object must be created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the AddCandidate method	Candidate	-1	-1	
2	Testing the AddCandidate method	Candidate	1	1	
3	Testing the AddCandidate method	Candidate	2	2	
4	Testing the RemoveCandidate method	0	0	0	
5	Testing the RemoveCandidate method	1	1	1	

Post condition(s) for Test:

- The number of candidates are set after removing or adding candidates.

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Add_Party
Test Description: This test will determine if the AddParty method correctly adds a unique party to the parties vector.

Name(s) of Testers: Emma Spindler

Automated: yes ✓ no

The tests are stored in the election_unittest.cpp file.
The methods used are AddParty, GetNumberOfParties

Results: Pass _✓_ Fail __

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **Must give a string name of the party**
- **An election object must created**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the AddParty method	"I"	1	1	
2	Testing the AddParty method with the same party	"I"	1	1	

Post condition(s) for Test:

- Unique parties are added to the parties vector

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-13-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Majority

Name(s) of Testers: Emma Spindler

Test Description: This test will test how the system will handle the election when there is a majority and when there is not, returning a whole number indicating the index of the winner if there is a majority and -1 if there isn't.

The tests are stored in the election_unittest.cpp file. The methods used are CheckForMajority, AddCandidate, SetNumberOfCandidates, IncreaseNumberOfCandidates, AddBallot, AddParty, SetVotesForParties, SetNumberOfBallots

Automated: yes ✓ no

Results: Pass _✓____ Fail_____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **Must be IR election**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the CheckForMajority method	none	0	0	
2	Testing the CheckForMajority method	none	-1	-1	
3	Testing the CheckForMajority method	none	2	2	
4	Testing the CheckForMajority method	none	-1	-1	

Post condition(s) for Test:

- The index of the majority winner is returned
- -1 is returned if there is not majority winner

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Tie
Test Description: This test will test that if the ResolveTie method returns a random number.

Name(s) of Testers: Emma Spindler

Automated: yes_✓__ no ___

The tests are stored in the election_unittest.cpp file.
The methods used are ResolveTie

Results: Pass _✓_____ Fail _____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the ResolveTie method	integer 3	random number	random number	

Post condition(s) for Test:

- A tie is resolved and random number indicating the winning index is returned

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_GetDateAndTime
Test Description: This test will test that if current time is given.

Name(s) of Testers: Emma Spindler

Automated: yes _✓__ no ___

The tests are stored in the election_unittest.cpp file.
The methods used are GetDateAndTime

Results: Pass ✓ Fail

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetDateAndTime method	none	current date and time	current date and time	

Post condition(s) for Test:

- The time is returned

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_UpdateBallotCurrDis
Test Description: This test will test that if the method accurately changes which candidate the ballot votes to the next candidate in order.

Name(s) of Testers: Emma Spindler

Automated: yes_✓__ no ___

The tests are stored in the election_unittest.cpp file.
The methods used are UpdateBallotCurrDis, AddCandidate, SetCurrDis, AddBallot, SetNumberOfCandidates

Results: Pass _✓_____ Fail _____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **the audit and media report must be already opened**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the UpdateBallotCurrDis method	Ballot	-1	-1	
2	Testing the UpdateBallotCurrDis method	Ballot	-1	-1	
3	Testing the UpdateBallotCurrDis method	Ballot	0	0	

Post condition(s) for Test:

- Changed which candidate the ballot votes to the next candidate in order.

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_Find_Candidate_To_Remove
Test Description: This test will test that if the FindCandidateToRemove method accurately finds the index of the candidate that has the lowest amount of votes.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file. The methods used are FindCandidateToRemove, SetNumberOfCandidates, AddCandidate, IncreaseNumberOfCandidates, SetVotesForParties, GetName, AddParty

Automated: yes_✓__ no ___

Results: Pass _✓_____ Fail_____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **the audit and media report must be already opened**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the FindCandidateToRemove method	none	0	0	
2	Testing the FindCandidateToRemove method	none	0	0	

3	Testing the FindCandidateToRemove method	none	0	0	

Post condition(s) for Test:

- Candidate is removed correctly

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_RedistributeBallot
Test Description: This test will test that if the method accurately distributes the ballots from the index of the candidate given to the remaining candidates.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file.
The methods used are RedistributeBallots, AddCandidate, SetCurrDis, UpdateBallotCurrDis, SetNumberCandidates, IncreaseNumberOfCandidates, GetBallotListSize, GetName

Automated: yes ✓ no

Results: Pass _✓____ Fail_____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **the audit and media report must be already opened**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the RedistrubteBallot method	1	1	1	
2	Testing the RedistrubteBallot method	1	1	1	

Post condition(s) for Test:

- Ballots are redistributed among remaining candidates.

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_GetCandidate
Test Description: This test will test that if the method accurately returns the candidate from a given index.

Name(s) of Testers: Emma Spindler

The tests are stored in the election_unittest.cpp file. The methods used are GetCandidate, SetNumberofCandidates, AddCandidate, IncreaseNumberOfCandidates, GetName

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **the audit and media report must be already opened**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the GetCandidate method	0	'Emma'	'Emma'	

Post condition(s) for Test:

- The correct candidate is returned

Project Name: Project 1: Voting System **Team# 11**

Test Date: 3-14-2021

Test Stage: Unit _✓_ System __

Test Case ID#: Election_CloseReports
Test Description: This test will test that both audit and media report files have been successfully saved and closed.

Name(s) of Testers: Emma Spindler

Automated: yes ✓ no

The tests are stored in the election_unittest.cpp file.
The methods used are CloseReports

Results: Pass _✓____ Fail_____

Preconditions for Test:

- **election.h and election.cpp must compile**
- **driver.h and driver.cpp must compile**
- **report.h and report.cpp must compile**
- **ballot.h and ballot.cpp must compile**
- **candidate.h and candidate.cpp must compile**
- **the audit and media report must be already opened**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Testing the CloseReports method	none	0	0	

Post condition(s) for Test:

- All report files are closed successfully.

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-14-2021

Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower, Emma Spindler

Test Case ID#: IR_100000_Candidates

Test Description: This test will run an instant runoff election with 100000 candidates and check if the system runs it in under 8 minutes.

The test is stored in ir100000.csv, audit_ir_100000.txt, media_ir_100000.txt, IR_execution_runtime.png. These files are stored in the ir100000 folder in the testing directory.

Automated: yes no ✓

Results: Pass __✓___ Fail_____

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file of 100000 ballots.	ir100000.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced	IR_execution_runtime.png shows that the system ran in under eight minutes and details of how the election went are in audit_ir_100000.txt .

Post condition(s) for Test:

- A media report is generated

- An audit report is generated
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-14-2021
Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower, Emma Spindler

Test Case ID#: OPL_100000_Candidates
Test Description: This test will run an OPL election with 100000 candidates and check if the system runs it in under 8 minutes

The test is stored in opl100000.csv, audit_opl_100000.txt, media_opl_100000.txt, OPL_execution_runtime.png. These files are stored in the opl100000 folder in the testing directory.

Automated: yes no ✓

Results: Pass __✓___ Fail_____

Preconditions for Test:
- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file of 100000 ballots.	opl100000.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced.	OPL_execution_runtime.png shows that the system ran in under eight minutes and details of how the election went are in audit_opl_100000.txt .

Post condition(s) for Test:
- A media report is generated
- An audit report is generated

- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-14-2021

Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower, Emma Spindler

Test Case ID#: IR

Test Description: This test will run a sample IR election and ensure the correct output was produced.

The test is stored in the ir directory with files ip_audit.txt, ir_media.txt, and ir.csv.

Automated: yes___ no__✓_

Results: Pass __✓___ Fail _____

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run on a sample ir election.	ir.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced.	

Post condition(s) for Test:

- A media report is generated
- An audit report is generated
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-14-2021

Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower, Emma Spindler

Test Case ID#: OPL

Test Description: This test will run a sample OPL election and ensure the correct output was produced.

The test is stored in the opl directory with files opl_audit.txt, opl_media.txt, and opl.csv.

Automated: yes___ no __✓_

Results: Pass __✓___ Fail

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run a sample opl election	opl.csv	Voting System executes successfully. Media and audit reports are produced	Voting System executes successfully and media and audit reports are produced	

Post condition(s) for Test:

- A media report is generated
- An audit report is generated
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_ **Test Date: 3-14-2021**

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

Test Description: This test will run to determine if the system randomly selects a winner in the event of a tie among candidates during an instant runoff election. The voting system is run twice with the same ballots to make sure that there is a different winner each time.

The test is stored in ir_tie1_audit.txt, ir_tie1_media.txt, ir_tie2_audit.txt, ir_tie2_media.txt, ir_tie.csv. These files are stored in the ir_tie folder in the testing directory.

Automated: yes no ✓

Results: Pass ✓ Fail

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file that would conclude in a tie.	ir_tie.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced with the winner being Kleinber ®.	
2	The same CSV file is run with the expected winner being different from last time.	ir_tie.csv	Voting System executes successfully. Media and audit reports are produced with the winner being someone other than Kleinberg (R).	Voting System executes successfully. Media and audit reports are produced with the winner being Chou (I).	

Post condition(s) for Test:

- 2 media reports generated with different results
- 2 audit reports are generated with different results
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-14-2021

Test Case ID#: OPL_Candidate_Tie

Name(s) of Testers: Hoai Bui, Ryan Mower

Test Description: This test will run to determine if the system randomly selects a winner in the event of a tie among candidates during an OPL election. The voting system is run twice with the same ballots to make sure that there is a different winner each time.

The test is stored in opl_candidate_tie1_audit.txt, opl_candidate_tie1_media.txt, opl_candidate_tie2_audit.txt, opl_candidate_tie2_media.txt, opl_candidate_tie.csv. These files are stored in the opl_candidate_tie folder in the testing directory.

Automated: yes___ no__✓_

Results: Pass __✓___ Fail_____

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file that would conclude in a tie.	opl_candidate_tie.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced with the winner being Smith (D)	
2	The same CSV file is run with the expected winner being different from last time.	opl_candidate_tie.csv	Voting System executes successfully. Media and audit reports are produced with the winner being someone other than Smith (D).	Voting System executes successfully. Media and audit reports are produced with the winner being Pike (D).	

Post condition(s) for Test:

- 2 media reports generated with different results
- 2 audit reports are generated with different results
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_ **Test Date: 3-14-2021**

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

Test Description: This test will run to determine if the system randomly selects a winner in the event of a tie among parties during an OPL election. The voting system is run twice with the same ballots to make sure that there is a different winner each time.

The test is stored in opl_party_tie1_audit.txt, opl_party_tie1_media.txt, opl_party_tie2_audit.txt, opl_party_tie2_media.txt, opl_party_tie.csv. These files are stored in the opl_party_tie folder in the testing directory.

Automated: yes___ no __✓_

Results: Pass ✓ Fail

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file that would conclude in a tie.	opl_party_tie.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced with the winner being Smith (D).	
2	The same CSV file is run with the expected winner being different from last time.	opl_party_tie.csv	Voting System executes successfully. Media and audit reports are produced with the winner being someone other than Smith (D).	Voting System executes successfully. Media and audit reports are produced with the winner being Pike (D).	

Post condition(s) for Test:

- 2 media reports generated with different results
- 2 audit reports are generated with different results
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_ **Test Date: 3-25-2021**

Test Case ID#: OPL_One
Test Description: This test will run an OPL election with 6 candidates and one ballot and check that it outputs the correct results.

Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower

The test is stored in opl_one.csv, opl_one_audit.txt, and opl_one_media.txt files. These files are stored in the opl_one folder of the testing directory.

Automated: yes___ no __✓_

Results: Pass __✓___ Fail _____

Preconditions for Test:
- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file of 1 ballot.	opl_one.csv	Voting System executes successfully. Media and audit reports are produced.	Voting System executes successfully. Media and audit reports are produced.	

Post condition(s) for Test:
- A media report is generated
- An audit report is generated
- Results are displayed on terminal

Project Name: Project 1: Voting System **Team#11**

Test Stage: Unit __ System _✓_

Test Date: 3-25-2021

Test Case ID#: IR_One

Test Description: This test will run an OPL election with 4 candidates and one ballot and check that it outputs the correct results.

Name(s) of Testers: Hoai Bui, Eric Palmer, Ryan Mower

The test is stored in ir_one.csv, ir_one_audit.txt, and ir_one_media.txt files. These files are stored in the ir_one folder of the testing directory.

Automated: yes___ no __✓_

Results: Pass __✓___ Fail _____

Preconditions for Test:

- **The voting system must compile**
- **There must be an appropriate CSV file to run**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The voting system is run with a CSV file of 1 ballot	ir_one.csv	Voting System executes successfully and media and audit reports are produced	Voting System executes successfully and media and audit reports are produced	

Post condition(s) for Test:

- A media report is generated
- An audit report is generated
- Results are displayed on terminal

Project 2 New Tests

System Tests

Testing Number: 001
PBI 5 Task 1: Write code to handle multiple CSV files for different election types.

Team Member(s) Responsible: Hoai Bui

Inputs:
1. ir.csv
2. ir.csv ir_one.csv
3. ir.csv ir_one.csv ir_simple.csv
4. opl.csv opl_2.csv
5. Invalid_File.csv

Tests:
1. Test for running the voting system with one CSV file corresponding to an IR election
2. Test for running the voting system with two CSV files corresponding to an IR election
3. Test for running the voting system with three CSV files corresponding to an IR election
4. Test for running the voting system with two CSV files corresponding to an OPL election
5. Test for running the voting system with an invalid CSV file name

Outputs: Election is successfully computed. Corresponding audit and media reports are created(can be found in the PBI5Task1 folder of the testing directory).

1.
```
bui00015@csel-vole3d-14:~/5801/repo-Team11/Project2/src$ ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
../misc/ir.csv
d
Invalid ballot 6
Compute IR election.
Number of candidates: 4
Number of ballots: 20
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
========== Round 1 ==========
Candidate Rosen now has 5 ballots.
Candidate Kleinberg now has 4 ballots.
Candidate Chou now has 5 ballots.
Candidate Royce now has 5 ballots.
========== Round 2 ==========
Candidate Rosen now has 5 ballots.
Candidate Chou now has 5 ballots.
Candidate Royce now has 9 ballots.
========== Round 3 ==========
Candidate Rosen now has 7 ballots.
Candidate Royce now has 12 ballots.
==========================================
Winning Candidate: Royce (L).
bui00015@csel-vole3d-14:~/5801/repo-Team11/Project2/src$
```

2.
```
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
../misc/ir.csv
../misc/ir_one.csv
d
Invalid ballot 6
Compute IR election.
Number of candidates: 4
Number of ballots: 20
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
========== Round 1 ==========
Candidate Rosen now has 5 ballots.
Candidate Kleinberg now has 5 ballots.
Candidate Chou now has 5 ballots.
Candidate Royce now has 5 ballots.
========== Round 2 ==========
Candidate Rosen now has 5 ballots.
Candidate Kleinberg now has 5 ballots.
Candidate Chou now has 10 ballots.
========== Round 3 ==========
Candidate Kleinberg now has 8 ballots.
Candidate Chou now has 12 ballots.
==========================================
Winning Candidate: Chou (I).
bui00015@csel-vole-42:~/5801/repo-Team11/Project2/src$
```

3.

```
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
../misc/ir.csv
../misc/ir_one.csv
../misc/ir_simple.csv
d
Invalid ballot 6
Compute IR election.
Number of candidates: 4
Number of ballots: 20
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
=========== Round 1 ===========
Candidate Rosen now has 5 ballots.
Candidate Kleinberg now has 5 ballots.
Candidate Chou now has 5 ballots.
Candidate Royce now has 5 ballots.
=========== Round 2 ===========
Candidate Rosen now has 5 ballots.
Candidate Chou now has 6 ballots.
Candidate Royce now has 9 ballots.
=========== Round 3 ===========
Candidate Chou now has 9 ballots.
Candidate Royce now has 11 ballots.
=========================================
Winning Candidate: Royce (L).
bui00015@csel-vole3d-14:~/5801/repo-Team11/Project2/src$
```

4.
```
src(master*) » ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
../testing/PBI5Task1/opl.csv
../testing/PBI5Task1/opl_2.csv
d
Compute OPL election.
Number of candidates: 6
Number of seats: 3
Number of ballots: 35
** Candidates **
Pike (D)
Foster (D)
Deutsch (R)
Borg (R)
Jones (R)
Smith (I)
=== Whole Number Seats ===
Party: D recieved 1 seats.
Party: R recieved 1 seats.
Party: I recieved 0 seats.
=== Remainder Seat Number ===
Party: D recieved 1 seats.
Party: R recieved 0 seats.
Party: I recieved 0 seats.
=== Total Seat Number ===
Party: D recieved 2 total seats.
Party: R recieved 1 total seats.
Party: I recieved 0 total seats.
====== WINNERS  ======
1. Pike (D)
2. Foster (D)
3. Borg (R)
```

5.
```
bui00015@csel-vole-20:~/5801/repo-Team11/Project2/src$ ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
Invalid_File.csv
d
File not valid. Invalid_File.csv
bui00015@csel-vole-20:~/5801/repo-Team11/Project2/src$
```

Passed or Failed: Passed

Date: 4-17-2021

Testing Number: 002
PBI 7 Task 1: Invalidate IRV ballots

Team Member(s) Responsible: Hoai Bui

Inputs:

1. ir1.csv
2. ir_bad.csv
3. ir_bad2.csv

Tests:

1. Test for running the election system with no invalid ballots
2. Test for running the election system when one ballot has less than half of the candidates ranked(ballot number 4)
3. Test for running the IR election system with two invalid ballots.

Outputs: Election is successfully computed with correct ballots invalidated. Corresponding audit and media reports are created(can be found in the PBI7Task1 folder of the testing directory).

1.

```
bui00015@csel-vole-29:~/5801/repo-Team11/Project2/src$ ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
ir1.csv
d
Compute IR election.
Number of candidates: 4
Number of ballots: 9
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
=========== Round 1 ===========
Candidate Rosen now has 0 ballots.
Candidate Kleinberg now has 0 ballots.
Candidate Chou now has 0 ballots.
Candidate Royce now has 9 ballots.
=========================================
Winning Candidate: Royce (L).
bui00015@csel-vole-29:~/5801/repo-Team11/Project2/src$
```

2.

```
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
ir_bad.csv
d
Invalid ballot 4
Compute IR election.
Number of candidates: 5
Number of ballots: 5
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
Bob (R)
=========== Round 1 ===========
Candidate Rosen now has 0 ballots.
Candidate Kleinberg now has 0 ballots.
Candidate Chou now has 1 ballots.
Candidate Royce now has 3 ballots.
Candidate Bob now has 0 ballots.
=========================================
Winning Candidate: Royce (L).
```

3.

```
src(master*) » ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
../testing/PBI7Task1/ir_bad2.csv
d
Invalid ballot 4
Invalid ballot 5
Compute IR election.
Number of candidates: 5
Number of ballots: 6
** Candidates **
Rosen (D)
Kleinberg (R)
Chou (I)
Royce (L)
Bob (R)
========== Round 1 ==========
Candidate Rosen now has 0 ballots.
Candidate Kleinberg now has 0 ballots.
Candidate Chou now has 1 ballots.
Candidate Royce now has 3 ballots.
Candidate Bob now has 0 ballots.
========== Round 2 ==========
Candidate Kleinberg now has 0 ballots.
Candidate Chou now has 1 ballots.
Candidate Royce now has 3 ballots.
Candidate Bob now has 0 ballots.
========== Round 3 ==========
Candidate Kleinberg now has 0 ballots.
Candidate Chou now has 1 ballots.
Candidate Royce now has 3 ballots.
========== Round 4 ==========
Candidate Chou now has 1 ballots.
Candidate Royce now has 3 ballots.
========== Round 5 ==========
=========================================
Winning Candidate: Royce (L).
```

Passed or Failed: Passed

Date: 4-22-2021

Testing Number: 003
PBI 3 Task 1: Load PO ballots

Team Member(s) Responsible: Hoai Bui

Inputs:

1. po.csv

Tests:

1. Test running the system on a CSV file for a PO election

Outputs: CSV file for PO election is read. Corresponding audit and media reports are created(can be found in the PBI3Task1 folder of the testing directory).

1.

```
bui00015@csel-vole-29:~/5801/repo-Team11/Project2/src$ ./votingsystem
Enter name of CSV file (or path to CSV file), or 'd' to be done entering names of CSVs.
po.csv
d
Ballot: 1 goes towards Pike (D)
Ballot: 2 goes towards Pike (D)
Ballot: 3 goes towards Foster (D)
Ballot: 4 goes towards Jones (R)
Ballot: 5 goes towards Smith (I)
Ballot: 6 goes towards Borg (R)
Ballot: 7 goes towards Borg (R)
Ballot: 8 goes towards Pike (D)
Ballot: 9 goes towards Foster (D)
Compute PO election.
Number of candidates: 6
Number of ballots: 9
** Candidates **
Pike (D)
Foster (D)
Deutsch (R)
Borg (R)
Jones (R)
Smith (I)
PO information read into memory.
bui00015@csel-vole-29:~/5801/repo-Team11/Project2/src$
```

Passed or Failed: Passed

Date: 4-22-2021

========== **Unit Tests** ==========

Testing Number: 004 Unit test for the constructor of the Driver class
Team Member(s) Responsible: Hoai Bui
Inputs: 1. “ballot1” “ballot2” “ballot3” 2. “ir.csv”
Tests: 1. Test for determining if the constructor properly handles a valid file without errors
Outputs: 1. “”
Passed or Failed: Passed
Date: 4-22-2021

Testing Number: 005 Unit test for the ReadInElectionType method
Team Member(s) Responsible: Eric
Inputs: 1. ir.csv 2. NULL
Tests: 1. Test for seeing if the ReadInElectionType properly handles a null pointer 2. Testing that the flag parameter works 3. Testing that the method works with a regular formatted file
Outputs: 1. "File Handle is not open for Election Type.\n" 2. “NONE” 3. “IR”

Passed or Failed: Passed
Date: 4-22-2021

Testing Number: 006 Unit test for the ReadInNumCandidates method
Team Member(s) Responsible: Eric
Inputs: 1. Null Pointer, 1 2. File Pointer , 1 3. File Pointer , 0
Tests: 1. Test for seeing if the ReadInNumCandidates properly handles a null pointer 2. Test for the case that there are no candidates read in. 3. Test for the case of multiple candidates being read in.
Outputs: 1. 0 2. -1 3. 4
Passed or Failed: Passed
Date: 04-27-2021

Testing Number: 007 Unit test for the ReadInCandidates method
Team Member(s) Responsible: Eric
Inputs: 1. NULL pointer, 1 2. File Pointer, 0

Tests:

1. See if ReadInCandidates properly handles a null pointer
2. Verify that ReadInCandidates properly updates the candidate names in the election
3. Verify that ReadInCandidates properly updates the candidate parties in the election

Outputs:

1. 0
2. “Chou”
3. “L”

Passed or Failed: Passed

Date: 04-27-2021

Testing Number: 008
Unit test for the ReadInNumberOfSeats method

Team Member(s) Responsible: Hoai Bui

Inputs:

1. opl.csv
2. NULL, 1

Tests:

1. Testing for when flag is 1
2. Testing for when fh is not a null pointer

Outputs:

1. 0
2. 3

Passed or Failed: Passed

Date: 04-27-2021

Testing Number: 009

Unit test for the ReadInNumBallots method
Team Member(s) Responsible: Ryan
Inputs: 1. Null pointer, 1 2. void, 9
Tests: 1. Reads in a null file, must ensure that the number of ballots read in are 0. 2. Reads in a normal file of 9 ballots, ensures all 9 ballots are read in.
Outputs: 1. 0 2. 9
Passed or Failed: Passed
Date: 04-27-2021

Testing Number: 010 Unit test for the ReadInBallots method
Team Member(s) Responsible: Eric
Inputs: 1. Null Pointer, 1 2. None
Tests: 1. Check that the ReadInBallots function properly handles invalid input 2. Check that the ReadInBallots function executes successfully when run on a valid ballot file
Outputs: 1. 0 2. 0

Passed or Failed: Passed
Date: 04-27-2021

Testing Number: 011 Unit test for the ParseLine method
Team Member(s) Responsible: Hoai Bui
Inputs: 1. "one, two, three, four"
Tests: 1. Testing to see if the ParseLine method correctly captures elements separated by commas in a string and appends them to a vector
Outputs: 1. “one” 2. “two” 3. “three” 4. “four”
Passed or Failed: Passed
Date: 4-27-21

Testing Number: 012 Unit test for the ParseLine2 method
Team Member(s) Responsible: Hoai Bui
Inputs: 1. "four, three, two, one"

Tests: 1. Testing to see if the ParseLine2 method correctly captures elements separated by commas in a string and appends them to a vector
Outputs: 1. “four” 2. “three” 3. “two” 4. “one”
Passed or Failed: Passed
Date: 4-27-21