Project Name: Project 1: Voting System **Team#7**

Test Stage: Unit __ System X **Test Date: 3/28/2023**

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

Test Description: System Tests for CPL

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

Automated: yes___ no X_ **src/gtest_code/CPLSysTests.cc**

Results: Pass X___ Fail________ **TEST_F, run_test(), main()**

Preconditions for Test:
Test successfully compiles and links with necessary object files

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	NormalCPSystemTest Tests a normal run with no advanced qualities in it	CPLNormalTest.csv	Foster, Green, McClure	Foster, Green, McClure	
2	TieNoRemainderCPLSystemTest Tests a tie case where there are 3 different parties each getting exactly the boundary of votes.	CPLNoRemainderTieTest.csv	Foster, Green , Jacks	Foster, Green, Jacks	
3	PureTieCPLSystemTest Tests a case where there is a tie among all parties and none reach the boundary	CPLPureTieTest	Not any second candidates, no same candidates	Not any second candidates, no same candidates	
4	WipeoutCPLSystemTest Tests a case where one party receives all the votes	CPLWipeoutTest.csv	Foster, Volz, Pike	Foster,Volz, Pike	

Post condition(s) for Test:

CPL running properly on various different scenarios and boundary scenarios

Project Name: Project 1: Voting System **Team#7**

Test Stage: Unit __ System X **Test Date: 3/28/2023**

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

Test Description: System Tests for IR

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

Automated: yes___ no X_

Results: Pass X___ Fail________ **TEST_F, run_test(), main()**

Preconditions for Test:
Test successfully compiles and links with necessary object files

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	IRNormalSystemTest Tests a normal run with no advanced qualities in it	IRNormalTest.csv	Rosen	Rosen	
2	WipeoutIRSystemTest Tests a case where one candidate receives all the votes	IRWipeoutTest.csv	Rosen	Rosen	
3	IRTieTestSystemTest Tests a case where there is an exact tie between 2 candidates	IRTieTest.csv	Rosen OR Kleinberg	Rosen OR Kleinberg	This test does not test IR's ability to evenly break a tie, but instead its ability to simply break a tie. IR's unit test tests its ability to evenly break a tie
4	IRMultipleElimTest Tests IR's ability to eliminate many candidates in an election with no clear winner and multiple candidates	IRMultipleElimTest.csv	Marcus	Marcus	

Post condition(s) for Test:

IR running properly on various different scenarios and boundary scenarios

Voting System (BallotTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/27/23
Test Case ID#: BallotTest_1	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: addChoice	**Description:** Add a choice to the ballot and have it returned with getChoice(). Call getChoice() again to make sure only 1 choice was added.
Automated: YES	**Test location:** Executable from ‘make BallotTest` or `make tests` compiles to /src/gtest_code/executables/BallotTest
Results: PASS	
Preconditions: Compile executable with `make BallotTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	b.addChoice(1) b.getChoice()	Ballot b (1)	1	1	getChoice removes the returned choice. b is now empty
2	b.getChoice()	Ballot b (empty)	-1	-1	getChoice returns -1 on error (when the ballot is empty)

Post condition(s) for Test:

Ballot b will be empty

Voting System (BallotTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/27/23
Test Case ID#: BallotTest_2	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: EmptyChoices	**Description:** Call getChoice on an empty ballot
Automated: YES	**Test location:** Executable from ‘make BallotTest` or `make tests` compiles to /src/gtest_code/executables/BallotTest
Results: PASS	
Preconditions: Compile executable with `make BallotTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	b.getChoice()	Ballot b (empty)	-1	-1	getChoice returns -1 on error (when the ballot is empty)

Post condition(s) for Test:

Ballot b will be empty

Voting System (BallotTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/27/23
Test Case ID#: BallotTest_3	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: addChoiceOrder	**Description:** add 10 random choices to a ballot using addChoice(). Test their expected values against an array of the same random data using getChoice()
Automated: YES	**Test location:** Executable from ‘make BallotTest` or `make tests` compiles to /src/gtest_code/executables/BallotTest
Results: PASS	
Preconditions: Compile executable with `make BallotTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Random number generation	Ballot b int arr[10]	Ballot b (a, b, … j), arr{a, b, …. j} *a-j are random numbers stored in the same order*	–	Unable to view private variables. arr should contain the same variables as b in the same ordering. This is tested in the next step

2	*for i=0 to i=10*
	b.getChoice()
		Ballot b
		int arr[10]
			b.getChoice() == arr[0]
			b.getChoice() == arr[1]
			…
			b.getChoice() == arr[9]
				b.getChoice = arr[0]
				b.getChoice = arr[1]
				…
				b.getChoice() = arr[9]	

Post condition(s) for Test:

Ballot b will be empty

Voting System (CandidateTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: CandidateTest_1	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: CandidateConstructor	**Description:** Test the constructor for the Candidate class
Automated: YES	**Test location:** Executable from ‘make CandidateTest` or `make tests` compiles to /src/gtest_code/executables/CandidateTest
Results: PASS	
Preconditions: Compile executable with `make CandidateTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	c = Candidate()	Candidate c	c.getName() == “” c.getParty() == “”	c.getName() == “” c.getParty() == “”	
2	c = Candidate(“mario”, “party”)	Candidate c	c.getName() == “mario” c.getParty() == “party”	c.getName() == “mario” c.getParty() == “party”	

Post condition(s) for Test:

Candidate C will be a new Candidate with name “mario” and party “party”

Voting System (CandidateTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: CandidateTest_2	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: getNumVotes	**Description:** Test the vote counting `getNumVotes()` method
Automated: YES	**Test location:** Executable from ‘make CandidateTest` or `make tests` compiles to /src/gtest_code/executables/CandidateTest
Results: PASS	
Preconditions: Compile executable with `make CandidateTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	c1.getNumVotes()	Candidate c1	0	0	c1 initialized in setup()
2	c1.addBallot(b1) c1.getNumVotes()	Candidate c1	1	1	

Post condition(s) for Test:

Candidate C will be a new Candidate with 1 ballot assigned to it

Voting System (CandidateTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: CandidateTest_3	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: removeBallot	**Description:** Test the removeBallot() function
Automated: YES	**Test location:** Executable from ‘make CandidateTest` or `make tests` compiles to /src/gtest_code/executables/CandidateTest
Results: PASS	
Preconditions: Compile executable with `make CandidateTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	c1.addBallot(b3)
	for i=1 to i=3
	c1.removeBallot(&b_temp)
		Candidate c1
		Ballot b_temp
			for int i=1 to i=3
			choice = b_temp.getChoice()
			c1.removeBallot(i, choice)
				for int i=1 to i=3
				choice = b_temp.getChoice()
				c1.removeBallot(i, choice)
					Test if removeBallot() returns an actual ballot.
					b3 is <1,2,3>

2	*add 3 ballots to c1’s ballots* c1.getNumVotes()	Candidate c	c.getNumVotes() == 3	c.getNumVotes() == 3	
3	c1.removeBallot()	Candidate c	c1.removeBallot() == 0 c.getNumVotes() == 2	c1.removeBallot() == 0 c.getNumVotes() == 2	remove ballot returns 0 on success
4	*remove all ballots from c1* c1.removeBallot()	Candidate c	c1.removeBallot() == 1	c1.removeBallot() == 1	remove ballot returns 1 on failure (no ballots)

Post condition(s) for Test:

Candidate c1 will have no ballots remaining

Voting System (CandidateTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/28/23**
Test Case ID#: CandidateTest_4	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: LoadTest	**Description:** Test the candidate class under load
Automated: **YES**	**Test location:** Executable from ‘make CandidateTest` or `make tests` compiles to /src/gtest_code/executables/CandidateTest
Results: **PASS**	
Preconditions: Compile executable with `make CandidateTest` or `make tests` from the /project1/ directory Makefile	

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	*for i=0 to i=100000* c1.addBallot()	Candidate c1	c1.getNumVotes() == 100000	c1.getNumVotes() == 100000	
2	*for i=0 to i=99999* result |= c1.removeBallot()	Candidate c int result	c.getNumVotes() == 1 result = 0	c.getNumVotes() == 1 result = 0	
3	*0 ballots left* result = c1.removeBallot()	Candidate c int result	result == 1	result == 1	remove ballot returns 1 on failure (no ballots)

Post condition(s) for Test:

Candidate c1 will have no ballots remaining

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: IRTest_1	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionNoCan	**Description:** Creates an empty vector of candidates and attempts to run the election.
Automated: YES	**Test location:** Executable from 'make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: PASS	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	std::vector<Candidate*> candidates = {}	candidates			Create an empty candidates vector
2	ir.runElection(candidates)	ir candidates	-1	-1	Election should fail if there are no candidates

Postcondition(s) for Test:

The election fails without candidates.

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: IRTest_2	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionOneCanOneBalEmpty	**Description:** Creates an empty vector of candidates and a ballot that is not assigned and attempts to run the election.
Automated: **YES**	**Test location:** Executable from ‘make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: **PASS**	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	candidates.push_back(new Candidate(name, party))	ir candidates	–	–	Creates a new candidate with name “Arnold” and party “Up”
2	Ballot b	b	–	–	Create a ballot without assigning
3	ir.runElection(candidates)	ir candidates	-1	-1	Election should fail without any ballots

Postcondition(s) for Test:

The election fails without ballots

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/28/23**
Test Case ID#: IRTest_3	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionOneCanOneBal	**Description:** Creates a candidates vector with one candidate and one ballot
Automated: **YES**	**Test location:** Executable from ‘make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: **PASS**	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	candidates.push_back(can1)	candidates can1	–	–	Creates new candidate with name “Arnold” and party “Up”
2	Ballot b	b	–	–	Create a ballot
3	candidates[0].addBallot(b)	candidates b	–	–	Assign the ballot to the candidate
4	ir.runElection(candidates)	ir candidates	0	0	Election should run successfully
5	Candidate winner = ir.getWinner()	ir winner	Winning candidate	Winning candidate	Calculates the winning candidate
6	winner.getName()	winner	“Arnold”	“Arnold”	The name of the winner should match “Arnold”

Postcondition(s) for Test:

The election runs successfully and finds the only candidate, Arnold, as the winner

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: IRTest_4	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionTwoCanTwoBal	**Description:** Creates a candidates vector with two candidates and two ballots
Automated: YES	**Test location:** Executable from ‘make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: PASS	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	candidates.push_back(new Candidate(name1, party1)	candidates	–	–	Creates new candidate with name “Arnold” and party “Up”
2	candidates.push_back(new Candidate(name2, party2)	candidates	–	–	Creates new candidate with name “Gerald” and party “Down”
3	Ballot b1	b1	–	–	Create a ballot
4	Ballot b2	b2	–	–	Create a ballot
5	candidates[0]->addBallot(b1)	candidates b1	–	–	Assign the ballot to candidate 1
6	candidates[0]->addBallot(b2)	candidates b2	–	–	Assign the ballot to candidate 1
7	ir.runElection(candidates)	ir candidates	0	0	Election should run successfully
8	Candidate winner = ir.getWinner()	ir winner	Winning candidate	Winning candidate	Calculates the winning candidate
9	winner.getName()	winner	“Arnold”	“Arnold”	The name of the winner should match “Arnold”

Postcondition(s) for Test:

The election runs successfully and finds the candidate with both ballots, Arnold, as the winner

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/28/23**
Test Case ID#: IRTest_5	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionThreeCanThreeBal	**Description:** Creates a candidates vector with three candidates and three ballots
Automated: **YES**	**Test location:** Executable from 'make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: **PASS**	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	candidates.push_back(new Candidate(name1, party1)	candidates	–	–	Creates new candidate with name "Arnold" and party "Up"
2	candidates.push_back(new Candidate(name2, party2)	candidates	–	–	Creates new candidate with name "Gerald" and party "Down"
3	candidates.push_back(new Candidate(name3, party3)	candidates	–	–	Create new candidate with name "Marvin" and party "Middle"
4	Ballot b1	b1	–	–	Create a ballot
5	Ballot b2	b2	–	–	Create a ballot
6	Ballot b3	b3	–	–	Create a ballot
7	candidates[0]->addBallot(b1)	candidates b1	–	–	Assign the ballot to candidate 1
8	candidates[1]->addBallot(b2)	candidates b2	–	–	Assign the ballot to candidate 2
9	candidates[1]->addBallot(b3)	candidates b3	–	–	Assign the ballot to candidate 2
10	ir.runElection(candidates)	ir candidates	0	0	Election should run successfully
11	Candidate winner = ir.getWinner()	ir winner	Winning candidate	Winning candidate	Calculates the winning candidate
12	winner.getName()	winner	"Gerald"	"Gerald"	The name of the winner should match "Gerald"

Postcondition(s) for Test:

The election runs successfully and finds the candidate with both ballots, Arnold, as the winner

Voting System (IRTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/28/23**
Test Case ID#: IRTest_6	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: IRRunElectionThreeCanThreeBal	**Description:** *****NOTE**: This test creates an extremely large test log, run at your own risk Creates a candidates vector with two candidates and four ballots to test a tie
Automated: **YES**	**Test location:** Executable from ‘make IRTest` or, `make tests` compiles to /src/gtest_code/executables/IRTest
Results: **PASS**	
Preconditions: Compile executable with `make IRTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	candidates.push_back(new Candidate(name1, party1)	candidates	–	–	Creates new candidate with name “Arnold” and party “Up”
2	candidates.push_back(new Candidate(name2, party2)	candidates	–	–	Creates new candidate with name “Gerald” and party “Down”
4	Ballot b1	b1	–	–	Create a ballot
5	Ballot b2	b2	–	–	Create a ballot
6	Ballot b3	b3	–	–	Create a ballot
7	Ballot b4	b4	–	–	Create a ballot
8	candidates[0]->addBallot(b1)	candidates b1	–	–	Assign the ballot to candidate 1
9	candidates[0]->addBallot(b2)	candidates b2	–	–	Assign the ballot to candidate 1
10	candidates[1]->addBallot(b3)	candidates b3	–	–	Assign the ballot to candidate 2
11	candidates[1]->addBallot(b4)	candidates b4	–	–	Assign the ballot to candidate 2
12	for (int i = 0; i < 10000; i++)	i	–	–	Loop to run the election 10,000 times
13	ir.runElection(candidates)	ir candidates	0	0	Election should run successfully
14	Candidate winner = ir.getWinner()	ir winner	Winning candidate	Winning candidate	Calculates the winning candidate
15	results.push_back(winner)	results	–	–	Each time the election is run, the winner is

		winner			calculated and that candidate is stored in the results vector
16	for (int i = 0; i < results.size(); i++)	results.size()	–	–	Loop through the results vector to test randomness
17	if (name1.compare(results[i].getName()) == 0) { c++ ; }	name1 results c	–	–	Check if “Arnold” is in the results vector - if it is, add 1 to c
18	c >= 4500 && c <= 5500	c	c >= 4500 && c <= 5500	c >= 4500 && c <= 5500	If the election is random, the amount of times “Arnold” wins the election should be between 4500 and 5500

Postcondition(s) for Test:

The election runs successfully 10000 times and finds that “Arnold” has won between 4500 and 5500 of them to demonstrate randomness

Voting System (CPLTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/29/23
Test Case ID#: CPLTest_1	**Name(s) of Testers:** Liam McGuigan (mcgui479)
Test Name: CPLTest	**Description:** Runs through each method and tests for the right inputs to functions and if there are outputting correctly as well.
Automated: YES	**Test location:** Executable from ‘make CPLTest` or `make tests` compiles to /src/gtest_code/executables/CPLTest
Results: PASS	
Preconditions: Compile the executable with `make CPLTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	int test= cpl->runElection	CPL cpl	test== 0	test == 0	The election runs without error
2	ReadBallotsTestType checks if readBallots() is taking a CPL ballot	cpl->readBallots(“CPLBallots.csv”)	cpl->getElectionTypeCP() == “CPL”	cpl->getElectionTypeCP() == “CPL”	Uses a getter to access the saved election type for an election
3	ReadBallotsTestNumParty Checks if readBallots() is reading in the correct number of parties from .csv	cpl->readBallots(“CPLBallots.csv”)	cpl->getNumParties() == 6	cpl->getNumParties() == 6	Uses a getter to access the saved number of parties for the read ballot
4	ReadBallotsTestPartyNames Checks if readBallots() is reading in the correct name of the parties from .csv	cpl->readBallots(“CPLBallots.csv”) vector<string> names1; names1 = cpl->getNames();	names1[0] = “Democratic” names1[1] = “Republican” names1[2] = “New Wave” names1[3] = “Reform” names1[4] = “Green” names1[5] = “Independent”	names1[0] = “Democratic” names1[1] = “Republican” names1[2] = “New Wave” names1[3] = “Reform” names1[4] = “Green” names1[5] = “Independent”	Uses a getter to access the saved names of parties for the read ballot and sets it equal to a

					vector in which I can check each index for the right name

5	CPLReadBallotsTestCandidates
	Checks if readBallots() is reading the correct candidates and storing them in the right party
		cpl->readBallots(“CPLBallots.csv”)	cpl->parties[0].getMembers()[0].getName() ="Foster";
			cpl->parties[0].getMembers()[1].getName() ="Volz";
			cpl->parties[0].getMembers()[2].getName() ="Pike";
			cpl->parties[1].getMembers()[0].getName() ="Green";
			cpl->parties[1].getMembers()[1].getName() ="Xu";
			cpl->parties[1].getMembers()[2].getName() = "Wang";
			cpl->parties[2].getMembers()[0].getName() = "Jacks";
			cpl->parties[2].getMembers()[1].getName() = "Rosen";
			cpl->parties[3].getMembers()[0].getName() ="McClure");
			cpl->parties[3].getMembers()[1].getName() = "Berg";
			cpl->parties[4].getMembers()[0].getName() = "Zheng";
			cpl->parties[4].getMembers()[1].getName() ="Melvin";
			cpl->parties[5].getMembers()[0].getNa
				cpl->parties[0].getMembers()[0].getName() ="Foster";
				cpl->parties[0].getMembers()[1].getName() ="Volz";
				cpl->parties[0].getMembers()[2].getName() ="Pike";
				cpl->parties[1].getMembers()[0].getName() ="Green";
				cpl->parties[1].getMembers()[1].getName() ="Xu";
				cpl->parties[1].getMembers()[2].getName() = "Wang";
				cpl->parties[2].getMembers()[0].getName() = "Jacks";
				cpl->parties[2].getMembers()[1].getName() = "Rosen";
				cpl->parties[3].getMembers()[0].getName() ="McClure");
				cpl->parties[3].getMembers()[1].getName() = "Berg";
				cpl->parties[4].getMembers()[0].getName() = "Zheng";
				cpl->parties[4].getMembers()[1].getName() ="Melvin";
				cpl->parties[5].getMembers()[0].getNa
					Uses the getMembers function for the party vector of cpl and then uses the getName() function from the candidate which is returned by as a member.

			me() ="Peters";	me() ="Peters";	
6	CPLReadBallotsTestNumSeats Tests if readBallots() is reading in the right number of seats and setting it to the right int	cpl->readBallots("CPLBallots.csv")	cpl->num_seats = 3	cpl->num_seats = 3	Accesses cpl's number of seats which is set during the readBallots using stoi
7	CPLReadBallotsTestNumBallots Checks if readBallots is reading in the right number of ballots from the line	cpl->readBallots("CPLBallots.csv")	cpl->getNumBallots() = 9	cpl->getNumBallots() = 9	Accesses a getter from cpl in which it returns the number of ballots read in during readBallots
8	CPLReadBallotsTestBallotTotals Checks if readBallots() is setting the right number of votes for each party	cpl->readBallots("CPLBallots.csv")	cpl->parties[0].getBallotTotal() = 3 cpl->parties[1].getBallotTotal() = 2 cpl->parties[2].getBallotTotal() = 0 cpl->parties[3].getBallotTotal() = 2 cpl->parties[4].getBallotTotal() = 1 cpl->parties[5].getBallotTotal() = 1	cpl->parties[0].getBallotTotal() = 3 cpl->parties[1].getBallotTotal() = 2 cpl->parties[2].getBallotTotal() = 0 cpl->parties[3].getBallotTotal() = 2 cpl->parties[4].getBallotTotal() = 1 cpl->parties[5].getBallotTotal() = 1	Accesses the party vector from cpl and uses the party function of getBallotTotal() which returns the total votes for each index of the party vector

9	CPLBreakTieTest	Candidate candidate("Foster", "Democratic");
		Candidate candidate1("Green", "Republican");
		CPL cpltie;
		Party democrat = new Party()
		Party republican = new Party()
		Int results[2];
			60 >=Result[0] >= 40
			60 >=Result[1] >= 40
				60 >=Result[0] >= 40
				60 >=Result[1] >= 40
					Ran a simulation of breaktie 100 times to make sure the random selection was choosing between the two parties about evenly

Post condition(s) for Test:

CPL cpl would have run a full election

Voting System (ElectionTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: ElectionTest_1	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: ElectionMakeAuditFileIRTest	**Description:** Attempts to create an audit file for an IR election
Automated: YES	**Test location:** Executable from 'make ElectionTest` or, `make tests` compiles to /src/gtest_code/executables/ElectionTest
Results: PASS	
Preconditions: Compile executable with `make ElectionTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	string name = election.makeAuditFile("IR")	name election	–	–	Creates an audit file and stores the audit file name as "name"
2	name.substr(0,3)	name	"IR_"	"IR_"	Checks the first three characters of the audit file to ensure it starts with "IR_"
3	name.size()	name	10	10	Checks the size of the audit file name, which should equal 10 (IR_MDDYYYY)

Postcondition(s) for Test:

The audit file is created for an IR election

Voting System (ElectionTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: ElectionTest_2	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: ElectionMakeAuditFileCPLTest	**Description:** Attempts to create an audit file for a CPL election
Automated: YES	**Test location:** Executable from ‘make ElectionTest` or, `make tests` compiles to /src/gtest_code/executables/ElectionTest
Results: PASS	
Preconditions: Compile executable with `make ElectionTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	string name = election.makeAuditFile(“CPL”)	name election	–	–	Creates an audit file and stores the audit file name as “name”
2	name.substr(0,3)	name	“CPL_”	“CPL_”	Checks the first four characters of the audit file to ensure it starts with “CPL_”
3	name.size()	name	11	11	Checks the size of the audit file name, which should be equal to 11 (CPL_MDDYYYY)

Postcondition(s) for Test:

The audit file is created for a CPL election

Voting System (ElectionTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: ElectionTest_3	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: ElectionWritetoAuditFileIRTest	**Description:** Attempts to write to an audit file for an IR election
Automated: YES	**Test location:** Executable from ‘make ElectionTest` or, `make tests` compiles to /src/gtest_code/executables/ElectionTest
Results: PASS	
Preconditions: Compile executable with `make ElectionTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	string name = election.makeAuditFile(“IR”)	name election	–	–	Creates an audit file and stores the audit file name as “name”
2	election.writeToAuditFile(“Testing, testing, testing”, name);	election name	–	–	Writes to the “name” audit file
3	getline(auditFile, line)	auditFile line	Testing, testing, testing	Testing, testing, testing	Reads the first line of the audit file & checks that the line in the file matches

Postcondition(s) for Test:

The audit file is written to in an IR election

Voting System (ElectionTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/28/23
Test Case ID#: ElectionTest_3	**Name(s) of Testers:** Justin Lau (lau00054)
Test Name: ElectionWritetoAuditFileCPLTest	**Description:** Attempts to write to an audit file for an CPL election
Automated: YES	**Test location:** Executable from ‘make ElectionTest` or, `make tests` compiles to /src/gtest_code/executables/ElectionTest
Results: PASS	
Preconditions: Compile executable with `make ElectionTest` or `make test` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	string name = election.makeAuditFile(“CPL”)	name election	–	–	Creates an audit file and stores the audit file name as “name”
2	election.writeToAuditFile(“Testing, testing, testing”, name);	election name	–	–	Writes to the “name” audit file
3	getline(auditFile, line)	auditFile line	Testing, testing, testing	Testing, testing, testing	Reads the first line of the audit file & checks that the line in the file matches

Postcondition(s) for Test:

The audit file is written to in an IR election

Voting System (PartyTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/29/23
Test Case ID#: PartyTest_1	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: BallotTotalManipulation	**Description:** Manipulate the ballotTotal class variable through the use of multiple getter / setter calls.
Automated: YES	**Test location:** Executable from ‘make PartyTest` or `make tests` compiles to /src/gtest_code/executables/PartyTest
Results: PASS	
Preconditions: Compile executable with `make PartyTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	p1.getBallotTotal()	Party p1 (no ballots)	ballotTotal == 0	ballotTotal == 0	Parties are initialized with 0 votes
2	p1.incBallotTotal()	Party p1 (1 ballot)	ballotTotal == 1	ballotTotal == 1	
3	*for i=0 to rand()* p1.incBallotTotal()	Party p1 (random # of ballots)	ballotTotal == rand() + 1	ballotTotal == rand() + 1	

Post condition(s) for Test:

Party p1 will have a random # of ballots assigned to it

Voting System (PartyTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/29/23**
Test Case ID#: PartyTest_2	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: SeatManipulation	**Description:** Manipulate the seatsWon variable and thoroughly test the winSeats() function
Automated: **YES**	**Test location:** Executable from 'make PartyTest` or `make tests` compiles to /src/gtest_code/executables/PartyTest
Results: **PASS**	
Preconditions: Compile executable with `make PartyTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	res = p1.winSeats(1)	Party p1 int res	seatsWon == 1 res == 0	seatsWon == 1 res == 0	winSeats() returns the number of extra seats assigned to the party
2	res = p1.winSeats(5)	Party p1 int res	seatsWon == 3 res = 3	seatsWon == 3 res == 3	p1 was initialized with 3 members. So the max for seatsWon is 3 and 3 extra seats are returned from winSeats
3	res = p1.winSeats(29)	Party p1 int res	seatsWon == 3 res = 29	seatsWon == 3 res = 29	p1 has three members. At this point in the test, any seats added will be extra so 29 is returned

Post condition(s) for Test:

Party p1 will have 3 seatsWon for its 3 members

Voting System (PartyTest) Team# 7

Test Stage: UNIT	**Test Date:** **3/29/23**
Test Case ID#: PartyTest_3	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: memberManipulation	**Description:** Manipulate the members of a Party and test the results with the members getters and setters
Automated: **YES**	**Test location:** Executable from ‘make PartyTest` or `make tests` compiles to /src/gtest_code/executables/PartyTest
Results: **PASS**	
Preconditions: Compile executable with `make PartyTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	p1.getMembers()	Party p1 <Candidate> Andretti	p1.getMembers() are the same as Andretti	p1.getMembers() are the same as Andretti	Andretti is initialized with candidates: c1 = “Colton Herta”, “Andretti” c2 = “Romain Grosjean”, “Andretti” c3 = “Kyle Kirkwood”, “Andretti” p1 has the same candidates added in setup()
2	Candidate c4 p1.addMember(c4)	Party p1 <Candidate> Andretti Candidate c4 = “Devlin Defranchesco”, “Andretti	p1.getMembers() are the same as the Andretti vector	p1.getMembers() are the same as the Andretti vector	Candidate c4 is added to the Andretti vector before comparison with p1.getMembers()
3	p2.getSeats()	Party p2	p2.getMembers() = 0	p2.getMembers() = 0	p2 is initialized as an empty party

Post condition(s) for Test:

Party p1 will have members c1 through c4, and p2 will not have any members

Voting System (PartyTest) Team# 7

Test Stage: UNIT	**Test Date:** 3/29/23
Test Case ID#: PartyTest_4	**Name(s) of Testers:** Lucas Olsen (olse0280)
Test Name: getWinners	**Description:** Test the functionality of the getWinners function
Automated: YES	**Test location:** Executable from ‘make PartyTest` or `make tests` compiles to /src/gtest_code/executables/PartyTest
Results: PASS	
Preconditions: Compile executable with `make PartyTest` or `make tests` from the /project1/ directory Makefile	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	p1.getWinners()	Party p1 <string> winners	winners.size() == 0	winners.size() == 0	Andretti is initialized with candidates: c1 = “Colton Herta”, “Andretti” c2 = “Romain Grosjean”, “Andretti” c3 = “Kyle Kirkwood”, “Andretti” p1 does not win any seats before this test

2	p1.winSeats(2)
	p1.getWinners()
		Party p1
		<string> winners
			winners.size() == 2, winners contains Candidates c1 and c2	winners.size() == 2, winners contains Candidates c1 and c2	2 seats are won, and winners are chosen from a party depending on the order in which members are added

3	p1.winSeats(99)
	p1.getWinners()
		Party p1
		<string> winners
			winners.size() == 3, winners contains c1, c2, and c3	p2.getMembers() = 0, winners contains c1, c2, and c3	

Post condition(s) for Test:

Party p1 will have members c1 through c3, and will have 3 seatsWon