Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit _X_ System __

Test Date: 03/24/23

Test Case ID#: 1
Test Description: IR Candidate Tests

Name(s) of Testers: Shivali Mukherji, Micheal Vang

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

Automated: yes_X__ no ___

Results: Pass ______ Fail__X_____

Preconditions for Test: test ballots of class Ballot needs to be created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	ConstructorTest	IRCandidate("name")	"name"	"name"	

3	serNumBallotsTest	setNumBallots(5)
		setNumBallots(1)
		setNumBallots(3)
			5
			1
			3
				5
				1
				3	

4	setBallotListTest	ballotList = {b1, b2, b3}	{b1, b2, b3}	{b1, b2, b3}	

5	addBallotTest	Ballot(1, {1,2,3})
		IRCandidate.addBallot(Ballot)
		test_candidate.getBallotList()
		test_candidate.getBallotList().back()
		test_candidate.getNumBallots()
			{b1, b2, b3, b4}
			.back() = {b4}
			numBallots = 4
				{b1, b2, b3, b4}
				.back() {b4}
				numBallots = 4
					Setter functions need to account for adding to the variable

		IRCandidate("Canid")
		B1 = (1, {1,2})
		B2 = (1. {2.1})
		Canid.addBallot(&B1);
		Canid.addBallot(&B2);
		Canid.getNumBallots()
		Canid.popBallot()
			numBallots = 2
			popped = {2,1}
			numBallots = 1
				numBallots = 2
				popped = {2,1}
				numBallots = 1	

6	addAndPopBallot				

Post condition(s) for Test: IRCandidate can be created with the ability to track the number of ballots, its mapping, and its name.

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 2

Name(s) of Testers: Shivali Mukherji

Test Description: IR Ballot Tests

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

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: methods must be validated in order for IRBallot object to be created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					

2	ConstructorTest	cans{c1,c2,c3,c4}
		s_map {1,2,3,4}
		map_% {0.25, 0.5, 0.75, 0.1}
		IRCandidate {cans, s_map, map_%)
			{c1,c2,c3,c4}
			{1,2,3,4}
			{0.25,0.5,0.75,0.1}
				{c1,c2,c3,c4}
				{1,2,3,4}
				{0.25,0.5,0.75,0.1}	

3	setGetCandidatesTest	setCandidates(),
		getCandidates()
		IRCandidates.size()
			numCandidates = 4
			IRCandidates = {c1,c2,c3,c4}
				candidates vector can be set and returned	

4	setMapPercentageTest	setMapPercentage(sample_map_percentage)
		ir_test_ballot.getMapPercentage()
			sample_map_percentage = {0.25, 0.5, 0.75, 0.1}	sample_map_percentage = {0.25, 0.5, 0.75, 0.1}	

5	setGetNumCandidatesTest	setNumCandidates(4), setNumCandidates(2), setNumCandidates(6), getNumCandidates()	4 2 6	4 2 6	

Post condition(s) for Test: IRBallot object is created and can be used throughout the program. The IRBallot will contain the map percentage for each candidate in a list, the number of ballots for each candidate in a list, and a list of candidates.

Test Stage: Unit X__ System __ **Test Date: 03/25/23**

Test Case ID#: 3 **Name(s) of Testers: Micheal Vang**

Test Description: Ballot Tests

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

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: methods must be validated in order for Ballot object to be created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	ConstructorTest	Ballot(1, {1,2,3,4})	Rank = 1 Mapping = {1,2,3,4}	Rank = 1 Mapping = {1,2,3,4}	

2	setInvalidRankingTest	.setRank(-1)
		.setRank(0)
		.setRank(5)
			rank = 1
			rank = 1
			rank = 4
				rank = 1
				rank = 1
				rank = 4	

3	setValidRankingTest	.setRank(1)
		.setRank(3)
		.setRank(4)
			rank = 1
			rank = 3
			rank = 4
				rank = 1
				rank = 3
				rank = 4	

4	increaseRankTest	.setRank(3)
		.increaseRank()
			rank = 4	rank = 4	

5	getAndSetIndex	.setRank(3) int index = .getIndex()	rank = 3 index = 2	rank = 3 index = 2	
6	ConstructorWithParameters	.getRank(), .getMapping(), vector<int> mapping{1, 2, 3}	rank = 1 Mapping = {1,2,3}	rank = 1 Mapping = {1,2,3}	
7	GetRank	vector<int> mapping{1, 2, 3}, .getRank()	rank = 1	rank = 1	
8	setGetMapping	std::vector<int> mapping1 = {1, 2, 3}; std::vector<int> mapping2	mapping = {4, 5, 6}	mapping = {4, 5, 6}	

		= {4, 5, 6};
		.setMapping(mapping2),
		.getMapping()
		Ballot(1, mapping1)			

Post condition(s) for Test: Ballot object is created and can be used throughout the program. The Ballot will containing a ranking for each candidate, as well as the mapping for each candidate.

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 4

Name(s) of Testers: Matin Horri

Test Description: AuditFile Tests

PBI 13 Correct Audit Name

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

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: methods must be validated in order for AuditFile object to be created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	ProductFileTest	open(),produceFile(), labelFile(“Test_File”), write(“line 1”)	“line 1”	“line 1”	
3	WriteTest_one	open(), write(“line 1”), write(“line 2”)	“line 1” “line 2”	“line 1” “line 2”	
4	DefaultConstructor	audit.getName()	“audit”	“audit”	
5	ConstructorWithArgs	audit(“test”)	“test”	“test”	
6	LabelFile	audit(“test”), labelFile(“new_test)	“new_test”	“new_test”	
7	Open	audit(“test”), open(), close()	“TRUE”	“TRUE”	
8	Close	audit(“test”), open(), close()	“FALSE”	“FALSE”	
9	ProduceFile	audit(“test”), open(), write(“Hello, World!”),	“Hello, World!”	“Hello, World”	

		produceFile(), Close(), file("test.txt")			
10	SetAndGetFile	fopent("test.txt), setFile(f),fclose(f)	"test.txt"	"test.txt"	
11	SetAndGetOutputResult	setOutputResult(test output), getOutputResult()	"test output"	"test output"	
12	SetAndGetFileName	setName(name), setFileName(filename), getName(), getFileName()	"test" "test.txt"	"test" "test.txt"	
13	GetFileStream	getFileStream() .good()	"TRUE"	"TRUE"	

Post condition(s) for Test: Auditfile object is created and can be used throughout the program. The Auditfile is produced after every election. This class allows the program to produce the file, label the file, and write to the file.

Test Stage: Unit X__ System __ Test Date: 03/25/23

Test Case ID#: 5
Test Description: CPLBallotTests

Name(s) of Testers: Wenjing Jiang

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

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: A file input has been given and CPLProcessing has processed the information

Tests	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	GetPartiesTests	“Democratic” “New Wave” test_cplParties[2]->getName() test_cplParties[0]->getName()	“Democratic” “New Wave”	“Democratic” “New Wave”	
2	getMapAllocatedSeatTests	mapAllocatedSeat = test_cplballot->getMapAllocatedSeat() mapAllocatedSeat[0] & [2]	0, 2,	0, 2	
3	setMapAllocatedSeatTest	int 1	1	1	
4	getMapRemainSeatTest	CPLBallot -> getMapRemainSeat()	mapRemainSeat[0] = 3 mapRemainSeat[2] = 2	3,2	
5	getSeatsTest	seat = CPLBallot->getSeats()	3	3	

6	setSeatsTest	test_cplBallot->setSeats(10);
		test_cplBallot->getSeats()
			10	10	

7	getMapBallotTest	CPLBallot -> getMapBallot() mapBallot[0] mapBallot[2]	mapBallot[0] = 3 mapBallot[2] = 0	3 0	
8	getNumPartiesTest	CPLBallot -> getNumParties()	6	6	
9	getQuotaTest	Ballot -> getQuota()	3	3	

Post condition(s) for Test: CPLBallot object holds the information correctly and can be used in a CPLVoteSystem.

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 6

Name(s) of Testers: Wenjing Jiang

Test Description: CPLProcessingTests

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

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: A file has been inputed that CPLProcessing can read

Tests	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	outputBallot	CPLProcess -> output() [CPLBallot] typeid(*cplBallot)	Type CPLBallot Object	Type CPLBallotObject	
2	inputsConsistency	CPLProcess -> readHead()	Succeed if the additional input file has same parties and candidates, and has CPL format	Succeed	
3	runMultipleFiles2	CPLProcess ->runMultipleFiles2()	Succeed if two input files are processed and have correct format	Succeed	
4	invalidFile	CPLProcess ->invalidFile()	Succeed if CPL3.csv has incorrect format and it's in the list of incorrect files	Succeed	

Post condition(s) for Test: CPLProcessing outputs a CPLBallot that can be used for CPLVoteSystem

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 7
Test Description: CPLVoteSystemTests

Name(s) of Testers: Wenjing Jiang

PBI 11 TIEBREAKER

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

Automated: yes_X__ no ___

Results: Pass ___X___ Fail_______

Preconditions for Test: methods must be validated in order for AuditFile object to be created

Tests	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	startElectionTest	CPLVoteSystem->StartElection	True	True	
2	conductElectionTest	CPLVoteSystem->ConductElection	True	True	
3	getWinnerTest	ConductElection() getWinner()	“Foster”	“Foster”	
4	remainSeatTest	getRemainSeat()	0	0	
5	CPLLotteryTest	CPLLottery(6)	Succeed if returned label smaller than 6	Succeed	
6	getWinnerWithMultipleFiles	getWinnerWithMultipleFiles()	“Foster”	“Foster”	

Post condition(s) for Test: CPLVoteSystem has successfully conducted its election and a winner is declared.

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 8

Name(s) of Testers: Matin Horri

Test Description: SpecialCase Tests

PBI 12 TIEBREAKER

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

Automated: yes X__ no ___

Results: Pass X Fail

Preconditions for Test: If there is no clear majority in IR between only two candidates, then popularity will win between the two. Also whenever there is a tie situation that has occurred between 2 things, i.e., a candidate or party, a fair coin will be tossed and determine who's the winner.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	tieBreakerValidGeneraor	t.run()	SUCCEED if tiebreaker result is less than 2	SUCCEED	
2	tieBreakerRunSize	t.run(3)	SUCCEED if tiebreaker result is less than 3	SUCCEED	
3	popularityCase	p.run(), "Tom", .getName()	winning candidate is "Tom"	winning candidate is "Tom"	
4	poptie	setNumBallots(20), p.run()	popularity case returns 100 + setNumBallots(20) if there is a tie	popularity case returns 100 + setNumBallots(20) if there is a tie	

Post condition(s) for Test: SpecialCase is implemented in cases where there was a tie situation that has occurred between 2 things in the CPL , and coin chose the winner, or popularity case for IR and popularity won the election between the two candidate.

Test Stage: Unit X__ System __

Test Date: 03/25/23

Test Case ID#: 9

Name(s) of Testers: Matin Horri

Test Description: Display Tests

NOT IMPLEMENTED IN PROGRAM

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

Automated: yes_X__ no ___

Results: Pass _X_______ Fail_________

Preconditions for Test: methods must be validated in order for Display object to be created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Overwrites	overWrite("test output",overWrite(“new output”), display.prtin()	“new output”	“new output”	
2	EmptyOutput	display.display(""), display.print()	“”	“”	
3	MultipleOverwrites	display.overWrite("first overwrite"), overWrtie (“second overwrite”), print()	“second overwrite”	“second overwrite”	
4	LongOutput	display.overWrite(large_output);	large_output	large_output	
5	EmptyOutput	`display.overWrite("");, display.print()`	SUCCEED if the result is an empty string	SUCCEED	
6	RandomOutput	display.overWrite(random_output), display.print()	random_ouptput	random_output	
7	OutputEquality	Display display1("test output"); Display display2("test output"); display1.print() display2.print()	“test output” “test output”	“test output” “test output”	

8	DefaulConstructor	Display display1(“”)	“ ”	“ ”	
9	InitializationConstructorTest	display(“Matin”)	“Matin”	“Matin”	
10	PrintTest	overwrite(“Matin”), print()	“Matin”	“Matin”	
11	GetOutputTerminalTest	display(“terminal”),getOutput Terminal()	“terminal1”	“terminal1”	
12	SetOutputTerminalTest	display(“Example”), setOutputTerminal(“Matin”), getOutputTerminal()	“Matin”	“Matin”	

Post condition(s) for Test: DisplayCase is implementedm, and the result is shown into the terminal.

Test Stage: Unit ___ System _X_

Test Date: 03/25/23

Test Case ID#: 10

Test Description: IR Vote System

Name(s) of Testers: Michael Vang

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

Automated: yes __X__ no ____

Results: Pass X Fail

Preconditions for Test: Test ballots must be created before running tests.

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	startElectionTest	ir_testVote->startElection()	expect true	true	
2	getWinnerTest	ir_testVote->getWinner(), Winner.getName()	expected result is “Rosen (D)”	“Rosen (D)”	
3	simpleGetSetWinner	IRCandidate("Johnny") setWinner(winner) ir_testVote->getWinner().getName(	“Johnny”	“Johnny”	
4	getSetProcessedBallot	Candidate John, Doe, Sally map_ballot {1,2,3} map_percentage{0.17, 0.33, 0.5}	IRBallot	IRBallot	
5	ranWithMultipleFiles	“data/IR.csv” “data/IR2.csv” “data/IR3.csv”	Success	Success	
6	winnerWithMultipleFiles	IR.csv IR2.csv	“Chou (I)”	“Chou (I)	

Post condition(s) for Test:
An election has been run for IR and the winner has been elected.

Test Stage: Unit __X__ System ___

Test Date: 03/25/23

Test Case ID#: 11

Name(s) of Testers: Michael Vang

Test Description: IR Processing

PBI 1.1 IR ELECTION

PBI 10 BUGFIX LINE READING

PBI 6.3 MULTI FILES

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

Automated: yes X no

Results: Pass _X___ Fail_____

Preconditions for Test: A file path must be provided for the test to run.

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	runSetup	"data/IR.csv"	expect true for setUp() function	true	
2	readFunction	"data/IR.csv"	expect true for setUp() & read()	true	setUp has to be used before read()
3	readCorrectly	"data/IR.csv"	expectedBallots = 6 expected # candidates = 4 expected candidates in vector	ballots = 6 numCan = 4 expected candidates in vector	
4	outputBallot	"data/IR.csv"	A type of IR Ballot	type of IR Ballot	
5	calculate	"data/IR.csv"	expected percentage map vector = formula calculated for percentage	expected percentage = formula calculated	
6	setGetFiles	"data/IR.csv" "data/IR2.csv"	files = {"data/IR.csv", "data/IR2.csv"}	files = {"data/IR.csv", "data/IR2.csv"}	
7	singularFile	"data/IR.csv"	Success()	Success()	
8	runMultipleFiles	"data/IR.csv" "data/IR2.csv"	Success()	Success()	
9	runInvalidFilesInMultiple	"data/IR.csv" "IR2.csv"	Success()	Success()	
10	getInvalidFiels	"data/IR.csv" "IR2.csv"	"IR2.csv"	"IR2.csv"	

11	mFilesCheckBallot	“data/IR.csv” “data/IR2.csv”	17	17	
12	runMultipleFiles3	IR.csv IR2.csv IR3.csv	Sucess	Sucess	
13	m3CheckBallot	IR.csv IR2.csv IR3.csv	24	24	

Post condition(s) for Test:
IRProcessing was able to process the provided file path.

Test Stage: Unit _X_ System ___

Test Date: 03/25/23
Name(s) of Testers: Wenjing Jiang, Micheal Vang

Test Case ID#: 12

Test Description: Party

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

Automated: yes__X__ no ____

Results: Pass __X___ Fail______

Preconditions for Test: Party class must be defined.

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setNameTest	testParty.setName(“Republican”)	“Republican”	“Republican”	
2	getNameTest	Foster, Volz, Democratic	Democratic	Democratic	
3	getCandidateTest	Foster, Volz	Foster	Foster	
4	setCandidateTest	Foster Volz	Volz	Volz	
5	getMaxSeatTest	getMaxSeat()	0	0	
6	setMaxSeatTest	setMaxSeat(2)	2	2	

Post condition(s) for Test:
Party class can be constructed that correctly contains information.

Test Stage: Unit _X_ System __

Test Date: **03/25/23**
Name(s) of Testers: Wenjing Jiang, Micheal Vang

Test Case ID#: 13

Test Description: Candidate

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

Automated: yes X no

Results: Pass __X___ Fail_______

Preconditions for Test: Candidate class must be defined.

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	.setName("Mike")	.setName("Mike")
		.getName()
		setParty("Demo")
		.getParty()
			name = Mike
			party = Demo
				name = Mike
				party = Demo	

2	constructer	Candidate("Dolly", "Repo")
		.getName()
		.getParty()
			name = Dolly
			party = repo
				name = Dolly
				party = repo	

Post condition(s) for Test:
Candidate class can be constructed and correctly hol information.

Test Stage: Unit _X_ System ___

Test Date: 04/28/23
Name(s) of Testers: Micheal Vang

Test Case ID#: 13

Test Description: TableBuilder

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

Automated: yes X no

Results: Pass __X___ Fail______

Preconditions for Test: TableBuilder class is defined

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	buildRow	T.buildrow()	suceess	success	
2	getSetRows	numRows = 5	numRows = 5	numRows = 5	
3	buildRowCheck	T.buildRow() 2 times	numRows = 2	numRows = 2	
4	tableAddError	addCell to empty table at row 1 & 2	empty.	empty	
5	tableAddError1	addCell to empty table at 0, -1, -2	empty	empty	
6	tableSetError1	setCell to empty table at row 4, 0, 1 w cell 1,1,2 respectively	empty	empty	
7	tableSetError2	build.Row setCell(0,1,"N/A" addCell(0,"N/A" setCell(0,2,"N/A)	succeed / no crash	succeed / no crash	

8	tableSetError3	setCell(-1,2,"N/A")
		setCell(0,-1,"N/A")
			succeed / no crash	suceed / no crash	

9	tableGetError1	getCell(2,4)
		getCell(0,1)
			succeed / no crash	succeed / no crash	

10	tableGetError2	getCell(-1,1)
		getCell(-2,-1)
			succeed / no crash	succeed / no crash	

11	table1	build table	Candidate Ballots Ross 10 Susan 15	Candidate Ballots Ross 10 Susan 15	
12	table2	build table w/ add cell	Candidate Ballots Ross (D) 10	Candidate Ballots Ross (D) 10	

			Susan (R) 15 Ralph (I)	Susan (R) 15 Ralph (I)	
13	table3	build table w/ setcell	Candidate Ballots Ross(D) 20 Alpha(R) 15 Ralph(I)	Candidate Ballots Ross(D) 20 Alpha(R) 15 Ralph(I)	
14	getCell	0, candidate 0, ballots	0,0 = Candidate 0, 1 = Ballots	0,0 = Candidate 0, 1 = Ballots	
15	getCellToInt	0, “candidate” 0, “10”	10	10	

Post condition(s) for Test:
A table can be constructed to hold data. Table can also be displayed and built.

Test Stage: Unit X__ System __

Test Date: 04/29/23

Test Case ID#: 14
Test Description: POProcessing Tests

Name(s) of Testers: Wenjing Jiang

PBI 2.1 PO Unit Tests

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

Automated: yes_X__ no ___

Results: Pass Fail X

Preconditions for Test: POProcessing class needs to be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	runSetUp	readHeader(0)	success	success	
2	outputBallot	output()	true	true	
3	getSetCandidate	setCandidates(candidates)	Sally (D)	Sally (D)	

		getCandidates()	Jane (R)	Jane (R)	

4	getSetMapPercentage	std::vector<double> percentages{0.25, 0.75, 0.4, 0.3}
		setMapPercentage(percentages)
		getMapPercentage()
			percentages = {0.25, 0.75, 0.4, 0.3}	percentages = {0.25, 0.75, 0.4, 0.3}	

5	readCorrectly	Candidate E1 = Candidate("Pike D", "D");
		Candidate E2 = Candidate("Foster D", "D");
		Candidate E3 = Candidate("Deutsch R", "R");
		Candidate E4 = Candidate("Borg R", "R");
		Candidate E5 = Candidate("Jone R", "R");
		Candidate E6 = Candidate("Smith I", "I");
		
		readHeader(0)
		read()
		getBLinesToRead()
		getNumCandidates()
			expectedBLines = 9
			expectedNumCanid = 6
			
			Pike D, D
			Foster D, D
			Deutsch R, R
			Borg R, R
			Jone R, R
			Smith I , I
				expectedBLines = 9
				expectedNumCanid = 6
				
				Pike D, D
				Foster D
				Deutsch R, R
				Borg R, R
				Jone R, R
				Smith I , I
					FAIL candidate name parsing wrong.

6	calculate	readHeader(0)
		read()
		std::vector<double> percentageMapping;
		std::vector<int> ballotMapping{3,2,0,2,1,1};
			percentageMapping = {0.333, 0.222, 0, 0.222, 0.111, 0.111}	percentageMapping = {0.333, 0.222, 0, 0.222, 0.111, 0.111}	**Fail here. Expected of system does not match actual**

Post condition(s) for Test: PoProcessing object can be created.

Test Stage: Unit _X_ System ___

Test Date: 04/29/23

Test Case ID#: 15

Test Description: POVoteSystem Tests

Name(s) of Testers: Michael Vang

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

Automated: yes __X__ no _____

Results: Pass Fail X

Preconditions for Test: Test ballots must be created before running tests.

Steps	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	startElectionTest	po_vote->startElection()	expect true	true	
2	getWinnerTest	po_vote->getWinner(), Winner.getName()	expected result is “Smith (I)”	“ Smith (I)”	**FAIL name of candidate error**
3	simpleGetSetWinner	POCandidate("Johnson I") setWinner(winner) po_vote->getWinner().getName()	“Johnson I”	“Johnson I”	
4	getSetProcessedBallot	Candidate John, Doe, Sally map_ballot {1,2,3} map_percentage{0.17, 0.33, 0.5}	POBallot	POBallot	

Post condition(s) for Test:

An election has been run for PO and the winner has been elected.

Test Stage: Unit X__ System __

Test Date: 04/29/23

Test Case ID#: 16

Name(s) of Testers: Micheal Vang

Test Description: Candidate Tests

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

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: Candidate class needs to be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setGetFunction	“Mike” “Demo”	“Mike” “Demo	“Mike” “Demo”	
2	constructer	Candidate(Dolly, Repo)	“Dolly” “Repo	“Dolly" “Repo”	

Post condition(s) for Test: Candidate object can be created. Classes can properly inherit from Candidate

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__ **Test Date: 03/29/23**

Test Case ID#: System1 **Name(s) of Testers: Micheal Vang**

Test Description: IR standard case Majority (ir2.csv)

PBI 9 IR Table

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/ir2.csv

Automated: yes____ no _X_

Results: Pass __X______ Fail __________

Preconditions for Test: testing/data/IR2.csv must exist and Main was be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main		Welcome screen prompt	Welcome screen prompt	
2	read testing/data/IR.csv		file gets processed and display the steps	file gets processed and display the steps	
3	Winner is given		Chou (I)	Chou (I)	
4	Auditfile produced		An audit file is produced with user prompt date	Audit file produced in same directory	
1.5	Run ./main testing/data/IR2.csv	IR2.csv	Stats / steps given and audit file produced IR Table produced Winner is Chou(I)	Stats / steps given and audit file produced in directory IR Table produced Winner is Chou(I)	

Post condition(s) for Test: IR election successfully conducted with the winner given and audit file produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__

Test Date: 04/29/23

Test Case ID#: System2

Name(s) of Testers: Micheal Vang

Test Description: IR Elimination w/ Pop case (IR.csv)

PBI 9 IR Table

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR2.csv

Automated: yes____ no _X_

Results: Pass X Fail

Preconditions for Test: testing/data/IR2.csv must exist and Main was be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR2.csv	IR2.csv	Stats / steps given and audit file produced in directory Candidate Kleinberg eliminated Candidate Royce eliminated Winner is Rosen IR Table produced	Stats / steps given and audit file produced in directory Candidate Kleinberg eliminated Candidate Royce eliminated Winner is Rosen IR Table produced	

Post condition(s) for Test: IR election successfully conducted with the case of elimination and redistribution. The winner is given and the audit file is produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__

Test Date: 03/29/23

Test Case ID#: System3

Name(s) of Testers: Micheal Vang

Test Description: IR Eliminate Tie (IR3.csv)

PBI 9 IR Table

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR3.csv

Automated: yes_____ no _X_

Results: Pass _X___ Fail_____

Preconditions for Test: testing/data/IR3.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR3.csv	IR3.csv	Stats / steps given and audit file produced in directory Tiebreaker is activated for candidate with lowest ballot. Candidate selected is either or. In this case, Kleinberg or Royce. IR Table produced Winner Rosen	Stats / steps given and audit file produced in directory "IRAudit.csv" Tiebreaker is activated for candidate with lowest ballot. Candidate selected is either or Kleinberg or Royce. IR Table produced Winner Rosen	./main testing/data/IR3.csv has been run multiple times in order to record that either or got eliminated.

Post condition(s) for Test: IR election successfully conducted with the case of tiebreaker of 2 people of candidate with the lowest votes. The winner is given and the audit file is produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__ **Test Date: 03/29/23**

Test Case ID#: System4 **Name(s) of Testers: Micheal Vang**

Test Description: IR Eliminate Tie 3 way (IR3way.csv)

PBI 9 IR Table

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR3way.csv

Automated: yes_____ no _X_

Results: Pass __X___ Fail____

Preconditions for Test: testing/data/IR3way.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR3way.csv	IR3way.csv	Stats / steps given and audit file produced in directory Tiebreaker is activated for candidate with lowest ballot. Candidate selected is either or out of three. IR Table produced Winner is Rosen(D)	Stats / steps given and audit file produced in directory "IRAudit.csv" Tiebreaker is activated for candidate with lowest ballot. Candidate selected is either or out of three. IR Table produced Winner is Rosen(D)	

Post condition(s) for Test: IR election successfully conducted with the case of tiebreaker of 3 candidates with the lowest votes. The winner is given and the audit file is produced.

Project Name: Project 1: Voting System — Team#24

Test Stage: Unit ___ System __X__

Test Date: 04/29/23

Test Case ID#: System5

Name(s) of Testers: Micheal Vang

Test Description: IR Popularity Tie Case (IR4.csv)

PBI 9 IR Table

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR4.csv

Automated: yes____ no _X_

Results: Pass X Fail

Preconditions for Test: testing/data/IR4.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR4.csv	IR4.csv	Stats / steps given and audit file produced in directory Popularity case is indicated for the winner Tiebreaker indicated for winner IR Table produced Winner is Rosen(I) or Chou(I)	Stats / steps given and audit file produced in directory Popularity case is indicated for the winner Tiebreaker indicated for winner IR Table produced Winner is Rosen(I) or Chou(I)	./main testing/data/IR4.csv has to be run multiple times in order to get either or.

Post condition(s) for Test: IR election successfully conducted with the case of popularity tiebreaker with audit file produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__

Test Date: 04/29/23

Test Case ID#: System6
Test Description: IR w/ Multiple Files (IR1.csv & IR2.csv) (IR1and2.csv to confirm)
PBI 6.1 6.2 6.3 9 MULTI

Name(s) of Testers: Micheal Vang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR.csv /testing/data/IR2.csv /testing/data/IR1and2.csv

Automated: yes no X

Results: Pass _X__ Fail___

Preconditions for Test: testing/data/IR.csv & IR2.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR.csv testing/data/IR2.csv	IR.csv IR2.csv	Stats / steps given and audit file produced in directory IR Table produced Winner is Chou(I)	Stats / steps given and audit file produced in directory IR Table produced Winner is Chou(I)	IR1and2.csv can be used instead of IR1 & IR2 to check.

Post condition(s) for Test: IR election successfully conducted with the case of 2 files with audit file produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__

Test Date: 04/29/23

Test Case ID#: System7

Name(s) of Testers: Micheal Vang

Test Description: IR w/ Multiple Files (IR1.csv & IR2.csv &IR3.csv)
PBI 6.1 6.2 6.3 9 MULTI

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
/testing/data/IR.csv /testing/data/IR2.csv /testing/data/IR3.csv

Automated: yes____ no _X_

Results: Pass _X__ Fail___

Preconditions for Test: testing/data/IR.csv & IR2.csv & IR3.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/IR.csv testing/data/IR2.csv testing/data/IR3.csv	IR.csv IR2.csv IR3.csv	Stats / steps given and audit file produced in directory IR Table produced Winner is Chou(I) 24 total ballots w/ 2 exhausted	Stats / steps given and audit file produced in directory IR Table produced Winner is Chou(I) 24 total ballots w/ 2 exhausted	

Post condition(s) for Test: IR election successfully conducted with the case of multiple files with audit file produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__ **Test Date: 03/29/23**

Test Case ID#: System8 **Name(s) of Testers: Micheal Vang**

Test Description: CPL (CPL.csv)

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

Automated: yes____ no _X_

Results: Pass _X_ Fail___

Preconditions for Test: testing/data/CPLLottery.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/CPL.csv	CPL.csv	Stats / steps given and audit file produced in directory "CPL.csv" Winner is Green, McClure, Peters (Republican, Reform, Independent)	Stats / steps given and audit file produced in directory "CPL.csv" Winner is Green, McClure, Peters (Republican, Reform, Independent)	

Post condition(s) for Test: Election for CPL is successfully conducted with the seats of the winner. An audit file is also produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__ **Test Date: 03/29/23**

Test Case ID#: System9 **Name(s) of Testers: Micheal Vang**

Test Description: CPLLottery (CPLLottery.csv)

Indicate where are you storing the tests (what file) and the name of the method/functions being used. /testing/data/CPL2.csv

Automated: yes____ no _X_

Results: Pass _X_ Fail___

Preconditions for Test: testing/data/CPL2.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/CPL2.csv	CPL2.csv	Stats / steps given and audit file produced in directory Winner is Foster(Democratic) The other 2 seats are randomly given to 2 different candidate/party.	Stats / steps given and audit file produced in directory “CPL.csv” Winner is Foster(Democratic) The other 2 seats are randomly given to 2 different candidate/party	

Post condition(s) for Test: Election for CPL is successfully conducted with the case of lottery. An audit file is also produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__

Test Date: 04/29/23

Test Case ID#: System10
Test Description: PO (PO.csv)
PBI 2.1 2.3

Name(s) of Testers: Wenjing Jiang

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

Automated: yes____ no _X_

Results: Pass X Fail

Preconditions for Test: testing/data/PO.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/PO.csv	PO.csv	Stats / steps given and audit file produced in directory "PO.csv" Winner is Smith (Independent)	Stats / steps given and audit file produced in directory "PO.csv" Winner is Smith (Independent)	

Post condition(s) for Test: Election for PO is successfully conducted. An audit file is also produced.

Project Name: Project 1: Voting System **Team#24**

Test Stage: Unit ___ System __X__ **Test Date: 04/29/23**

Test Case ID#: System11 **Name(s) of Testers: Wenjing Jiang**

Test Description: PO TieBreaker Case (PO2.csv)

PBI PBI 2.2 2.3

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

Automated: yes____ no _X_

Results: Pass X Fail

Preconditions for Test: testing/data/PO2.csv must exist and Main compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./main testing/data/PO2.csv	PO2.csv	Stats / steps given and audit file produced in directory “PO2.csv” Winner is either Borg (Republican) or Pike (Democratic) based on tiebreaker result	Stats / steps given and audit file produced in directory “PO2.csv” Winner is either Borg (Republican) or Pike (Democratic) based on tiebreaker result	

Post condition(s) for Test: Election for PO2 is successfully conducted in the case of a tiebreaker. An audit file is also produced.