Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP1 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the CPLParty Constructor

setup with:
```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, ConstructorTest
Functions: CPLParty()

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: Accurate types passed into constructor

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new CPLParty object	party candidates			
2	test party name	->get_name()	Party 1	Party 1	
3	get the candidates vect	party->get_candidates()			
4	check size of each vector		3	3	
5	check equality of Candidate objects		all objects equal	all objects equal	

Post condition(s) for Test: party is properly instantiated

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP2 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the get_candidates function

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, GetCandidatesTest
Functions: get_candidates(), get_name

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new CPLParty object	party candidates			
2	get the candidates vect	partyCandidates = party->get_candidates()			
3	check size of candidates vectors		3	3	
4	Check that candidate in party matches name of “Candidate 1”	partyCandidates	Candidate 1	Candidate 1	
5	Check that candidate in party matches name of “Candidate 2”	partyCandidates	Candidate 2	Candidate 2	
6	Check that candidate in party matches name of “Candidate 3”	partyCandidates	Candidate 3	Candidate 3	

Post condition(s) for Test: get_candidates properly returns candidate vector

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP3 **Name(s) of Testers: Grant Oie**

Test Description:

Tests get_name()

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, GetNameTest
Functions: get_name()

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	instantiate new CPLParty object	party candidates			
2	test party name	->get_name()	Party 1	Party 1	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP4 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the set_total_votes and get_total_votes function

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, GetSetTotalVotes
Functions: get_total_votes, set_total_votes

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new CPLParty object	party candidates			
2	set total votes to 5	party			
3	get total votes, check equal to 5	party	5	5	
4					
5					

Post condition(s) for Test: party is properly instantiated

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP5 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the assignseatwinner on party w/ 3 candidates, 2 seats

setup with:
```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, AssignSeatWinners_base
Functions: assign_seat_winners(seats)

Automated: yes X no

Results: Pass ___X_ Fail________

Preconditions for Test: seats parameter >= 0

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new CPLParty object	party candidates			
2	call assign seatwinners(2)				
3	check that first two candidates got winner designation		true	true	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP6 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the assignseatwinner on party w/ 3 candidates, 0 seats

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, AssignSeatWinners_zero
Functions: assign_seat_winners(seats)

Automated: yes X no

Results: Pass ___X_ Fail________

Preconditions for Test: seats parameter >= 0

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	instantiate new CPLParty object	party candidates			
2	call assign seatwinners(0)				
3	check that no candidates were assigned seat		true	true	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: CP7 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the assignseatwinner on party w/ 3 candidates, 4 seats

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new CPLCandidate("Candidate 1"));
 candidates.push_back(new CPLCandidate("Candidate 2"));
 candidates.push_back(new CPLCandidate("Candidate 3"));
```

Filename: CPLPartyUnitTest.cpp
Testname: CPLPartyTest, AssignSeatWinners_over
Functions: assign_seat_winners(seats)

Automated: yes X no

Results: Pass ___X_ Fail________

Preconditions for Test: seats parameter >= 0

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new CPLParty object	party candidates			
2	call assign seatwinners(4)				
3	check that all candidates were assigned seat		true	true	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP1 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the OPLParty Constructor

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, ConstructorTest
Functions: OPLParty()

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: Accurate types passed into constructor

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	instantiate new OPLParty object	party candidates			
2	test party name	->get_name()	Party 1	Party 1	
3	get the candidates vect	party->get_candidates()			
4	check size of each vector		3	3	
5	check equality of Candidate objects		all objects equal	all objects equal	

Post condition(s) for Test: party is properly instantiated

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP2 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the get_candidates function

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, GetCandidatesTest
Functions: get_candidates(), get_name

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new OPLParty object	party candidates			
2	get the candidates vect	partyCandidates = party->get_candidates()			
3	check size of candidates vectors		3	3	
4	Check that candidate in party matches name of “Candidate 1”	partyCandidates	Candidate 1	Candidate 1	
5	Check that candidate in party matches name of “Candidate 2”	partyCandidates	Candidate 2	Candidate 2	
6	Check that candidate in party matches name of “Candidate 3”	partyCandidates	Candidate 3	Candidate 3	

Post condition(s) for Test: get_candidates properly returns candidate vector

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP3 **Name(s) of Testers: Grant Oie**

Test Description:

Tests get_name()

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, GetNameTest
Functions: get_name()

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new OPLParty object	party candidates			
2	test party name	->get_name()	Party 1	Party 1	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP4 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the set_total_votes and get_total_votes function

setup with:
```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, GetSetTotalVotes
Functions: get_total_votes, set_total_votes

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate new OPLParty object	party candidates			
2	set total votes to 5	party			
3	get total votes, check equal to 5	party	5	5	
4					
5					

Post condition(s) for Test:

party is properly instantiated

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_   System __ **Test Date: 3/24/24**

Test Case ID#: OP5 **Name(s) of Testers: Grant Oie**

Test Description:

Tests the num_votes attribute, and calculate_total_votes, get_total_votes functions

calculate_total_votes is called in constructor and not immediately visible

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, GetTotalVotesTest
Functions: ->num_votes, get_total_votes, calculate_total_votes

Automated: yes_X_   no ___

Results: Pass ___X_   Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assign num_votes to candidates in candidates vector	candidates			
2	instantiate new OPLParty object	party candidates			
3	check that party.get_total_votes == sum of votes assigned to candidates		60	60	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP6 **Name(s) of Testers: Grant Oie**

Test Description:
Tests assign_seat_winners with 1 seat and a clear winner and get_winner functions

setup with:
```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest,AssignSeatWinners_singleSeat
Functions: assign_seat_winners, get_winner

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: XXX

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assign num_votes to candidates in candidates vector: 30, 20, 10 respectively	candidates			
2	instantiate new OPLParty object	party candidates			
3	call party->assign_seat_winners(1)	party			
4	Assert_true(candidate[0]->get_winner())		true	true	
5	Assert_true(candidate[1]->get_winner())		false	false	
6	Assert_true(candidate[2]->get_winner())		false	false	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP7 **Name(s) of Testers: Grant Oie**

Test Description:
Tests assign_seat_winners with 2 seat and a clear winners and get_winner functions

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest,AssignSeatWinners_multipleSeats
Functions: assign_seat_winners, get_winner

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: XXX

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assign num_votes to candidates in candidates vector: 30, 25, 20 respectively	candidates			
2	instantiate new OPLParty object	party candidates			
3	call party->assign_seat_winners(2)	party			
4	Assert_true(candidate[0]->get_winner())		true	true	
5	Assert_true(candidate[1]->get_winner())		true	true	
6	Assert_true(candidate[2]->get_winner())		false	false	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: OP8 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the assign_seat_winners() function under a tie-breaker scenario. 3-way tie for 2 seats. Ran several times to verify that the same candidates are not being selected each time.

setup with:

```
 std::vector<Candidate*> candidates;
 candidates.push_back(new OPLCandidate("Candidate 1"));
 candidates.push_back(new OPLCandidate("Candidate 2"));
 candidates.push_back(new OPLCandidate("Candidate 3"));
```

Filename: FileOPLPartyUnitTest.cpp
Testname: OPLPartyTest, AssignSeatWinners_equalVotes
Functions: assign_seat_winner

Automated: yes X no

Results: Pass ___X_ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assign num_votes to candidates in candidates vector: 30 for all	candidates			
2	instantiate new OPLParty object	party candidates			
3	call party->assign_seat_winners(2)	party			
4	sum number of winners in				

	candidate array				
5	check num winners		2	2	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP1 **Name(s) of Testers: Grant Oie**

Test Description:
test tokenize_lines against, base case

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, TokenizeLines_base
Functions: tokenize_lines

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	initialize string	"Token1,Token2,Token3"			
2	call tokenize_lines on string	vector<string>			
3	check size of vector		3	3	
4	check equality of vector[0]		“Token1”	“Token1”	
5	check equality of vector[1]		“Token2”	“Token2”	
6	check equality of vector[2]		“Token3”	“Token3”	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP2 **Name(s) of Testers: Grant Oie**

Test Description:
test tokenize_lines with whitespaces added

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, TokenizeLines_whitespaces
Functions: tokenize_lines

Automated: yes_X_ no ___

Results: Pass ___X_ Fail__________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	initialize string	"Token1 , Token2,Token3 "			
2	call tokenize_lines on string	vector<string>			
3	check size of vector		3	3	
4	check equality of vector[0]		“Token1”	“Token1”	
5	check equality of vector[1]		“Token2”	“Token2”	
6	check equality of vector[2]		“Token3”	“Token3”	

Post condition(s) for Test:

Project Name: Voting System

Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP3

Name(s) of Testers: Grant Oie

Test Description:
test tokenize_lines on a ballot example

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, TokenizeLines_ballots1
Functions: tokenize_lines

Automated: yes _X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	initialize string	",,1, "			
2	call tokenize_lines on string	vector<string>			
3	check size of vector		4	4	
4	check equality of vector[0]		“”	“”	
5	check equality of vector[1]		“”	“”	
6	check equality of vector[2]		“1”	“1”	
7	check equality of vector[3]		“”	“”	

Post condition(s) for Test:

Project Name: Voting System　　　Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP4

Name(s) of Testers: Grant Oie

Test Description:

test tokenize_lines on a ballot example

setup with:

```
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;
```

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, TokenizeLines_ballots2
Functions: tokenize_lines

Automated: yes_X_ no ___

Results: Pass ___X_ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	initialize string	"1,,, "			
2	call tokenize_lines on string	vector<string>			
3	check size of vector		4	4	
4	check equality of vector[0]		"1"	"1"	
5	check equality of vector[1]		""	""	
6	check equality of vector[2]		""	""	
7	check equality of vector[3]		""	""	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP5 **Name(s) of Testers: Grant Oie**

Test Description:

test tokenize_lines on a ballot example

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, TokenizeLines_ballots3
Functions: tokenize_lines

Automated: yes X no

Results: Pass ___X_ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	initialize string	",,,1 "			
2	call tokenize_lines on string	vector<string>			
3	check size of vector		4	4	
4	check equality of vector[0]		“”	“”	
5	check equality of vector[1]		“”	“”	
6	check equality of vector[2]		“”	“”	
7	check equality of vector[3]		“1”	“1”	

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP6 **Name(s) of Testers: Grant Oie**

Test Description:
test the accuracy of create_OPL_candidates function

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, CreateOPLCandidates
Functions: create_OPL_candidates

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: there are candidates in the csv file

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	open opltestfile				
2	check that file opened properly				
3	read first line into electionType	electionType			
4	read second line into numSeats	numSeats			
5	read third line into numBallots	numBallots			
6	read fourth line into numCandidates	numCandidates			
7	call create_OPL_candidates(file, numCandidates)	candidates = vector <tuple<partyname, candidate*>>			
8	check equality of numSeats, numBallots, numCandidates, candidates.size() against data in file		true	true	
9	check each partyname and candidate's name in the candidates vector, calling ->get_name() on each candidate and expect_eq'ing against the relevant name in the csv file		true	true	
10	delete allocated candidate memory				

Post condition(s) for Test:

Project Name: Voting System

Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP7

Name(s) of Testers: Grant Oie

Test Description:

test the accuracy of assign_votes_to_candidates function

setup with:

```
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
```

```
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;
```

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, AssignVotesToCandidates
Functions: create_OPL_candidates,assign_votes_to_candidates

Automated: yes _X_ no

Results: Pass ___X_ Fail________

Preconditions for Test: there are candidates in the csv file, there are ballots in the csv file

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	open opltestfile				
2	check that file opened properly				
3	read first line into electionType	electionType			
4	read second line into numSeats	numSeats			
5	read third line into numBallots	numBallots			
6	read fourth line into numCandidates	numCandidates			
7	call create_OPL_candidates(file, numCandidates)	candidates = vector <tuple<partyname, candidate*>>			
8	extract just candidates from the vector<tuple>>	candidatesVec			
9	call assign_votes_to_candidates(file,candidatesVec)				
10	close file				
11	check equality of candidates[i]->get_num_votes() against the expected vote value		true	true	
12	delete allocated candidate memory				

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP8 **Name(s) of Testers: Grant Oie**

Test Description:

test the accuracy of create_opl_parties function

setup with:

```
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;
```

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, CreateOPLParties
Functions: create_OPL_candidates,create_OPL_Parties

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: there are candidates in the csv file

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	open opltestfile				
2	check that file opened properly				

3	read first line into electionType	electionType			
4	read second line into numSeats	numSeats			
5	read third line into numBallots	numBallots			
6	read fourth line into numCandidates	numCandidates			
7	call create_OPL_candidates(file, numCandidates)	candidates = vector <tuple<partyname, candidate*>>			
8	call create_OPL_parties(candidates)	parties			
9	call assign_votes_to_candidates(file,candidatesVec)				
10	check parties size		3	3	
11	check parties name and size of candidate vector against expected values		true	true	
12	delete allocated party and candidate memory				

Post condition(s) for Test:

Project Name: Voting System

Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP9

Name(s) of Testers: Grant Oie

Test Description:

test the accuracy of create_cpl_parties function

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, CreateCPLParties
Functions:create_CPL_Parties

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: there are parties/candidates in the csv file

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	open cpltestfile				
2	check that file opened properly				
3	read first line into electionType	electionType			
4	read second line into numSeats	numSeats			
5	read third line into numBallots	numBallots			
6	read fourth line into numCandidates	numCandidates			
7	call create_CPL_parties(file, numCandidates)	parties = vector <Party*>			
8	close file				
9	check parties size		6	6	
10	check party name and size of candidate array against expected values		true	true	
11	delete allocated party memory				

Post condition(s) for Test:

Project Name: Voting System

Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP10

Name(s) of Testers: Grant Oie

Test Description:

test the accuracy of assign_votes_to_parties function

setup with:
oplTestFile = "../testing/test_data/sys_test1_opl.csv";
cplTestFile = "../testing/test_data/sys_test3_cpl.csv";
std::string oplTestFile;
std::string cplTestFile;
std::string election_type;
int numSeats;
int numBallots;
std::string line;
int numCandidates;
int numParties;

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, AssignVotesToParties
Functions:create_CPL_parties, assign_votes_to_parties

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test: there are parties and votes in the csv file

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	open cpltestfile				
2	check that file opened properly				
3	read first line into electionType	electionType			
4	read second line into numSeats	numSeats			
5	read third line into numBallots	numBallots			
6	read fourth line into numCandidates	numCandidates			
7	call create_CPL_parties(file, numCandidates)	parties = vector <Party*>			
8	call assign_votes_to_parties(file,parties)				
9	close file				
10	check parties vote counts		true	true	

	against expected values				
11	delete allocated party memory				

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP11 **Name(s) of Testers: Grant Oie**

Test Description:
test that ElectionDataParser properly creates CPL election

Filename: ElectionDataParserUnitTest.cpp
Testname: CPLFilesTest
Functions: create_election

Automated: yes_X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with single cpl election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 4/18/24**

Test Case ID#: EDP12 **Name(s) of Testers: Grant Oie**

Test Description:
test that ElectionDataParser properly creates OPL election

Filename: ElectionDataParserUnitTest.cpp
Testname: OPLFilesTest
Functions: create_election

Automated: yes _X_ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with single opl election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Voting System Team# 13

Test Stage: Unit _X_ System __ **Test Date: 4/18/24**

Test Case ID#: EDP13 **Name(s) of Testers: Grant Oie**

Test Description:
test that ElectionDataParser properly creates MPO election

Filename: ElectionDataParserUnitTest.cpp
Testname: MPOFilesTest
Functions: create_election

Automated: yes _X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with single mpo election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP14 **Name(s) of Testers: Grant Oie**

Test Description:
test that ElectionDataParser properly creates CPL election with multiple files

Filename: ElectionDataParserUnitTest.cpp
Testname: CPLMultipleFilesTest
Functions: create_election

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with multiple cpl election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Voting System

Team# 13

Test Stage: Unit _X_ System __

Test Date: 3/24/24

Test Case ID#: EDP15

Name(s) of Testers: Grant Oie

Test Description:
test that ElectionDataParser properly creates OPL election with multiple files

Filename: ElectionDataParserUnitTest.cpp
Testname: OPLMultipleFilesTest
Functions: create_election

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with multiple opl election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/24/24**

Test Case ID#: EDP15 **Name(s) of Testers: Grant Oie**

Test Description:
test that ElectionDataParser properly creates MPO election with multiple files

Filename: ElectionDataParserUnitTest.cpp
Testname: MPOMultipleFilesTest
Functions: create_election

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create election with multiple mpo election test file	ElectionData* election			
2	check that election!= nullptr		True	True	
3					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: OC 1 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests OPLCandidate get_name() return value.

Repo-Team13/Project1/testing/OPLCandidateUnitTest.cpp

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: OPLCandidate compiles and a Candidate object is initialized correctly.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create Candidate object	OPLCandidate* joe = new OPLCandidate(“Joe Schmo”)	None	None	Candidate joe is initialized
3	Compare get_name to expected result	EXPECT_EQ(“Joe Schmo”, joe->get_name());	True	True	joe->get_name() returns Joe Schmo
4	Create Candidate object	OPLCandidate* sam = new OPLCandidate(“Sam Politician”)	None	None	Candidate sam is initialized
5	Compare get_name to expected result	EXPECT_EQ(“Sam Politician”, sam->get_name());	True	True	sam->get_name() returns Sam Politician

Post condition(s) for Test:

Get_name correctly outputs a Candidate’s name. It can be used in the to_string function.

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: OC 2 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests OPLCandidate get_winner() return value.

repo-Team13/Project1/testing/OPLCandidateUnitTest.cpp

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: OPLCandidate compiles and a Candidate object is initialized correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Candidate steven initialized	OPLCandidate* steven = new OPLCandidate ("Steven Carter");	Candidate initialized	Candidate initialzed	Candidate initialized
3	Compare winner variable with default winner value of false	EXPECT_EQ(false, steven->get_winner());	True	True	Candidate initialized with a winner value of false
4	Set winner to true	steven->set_winner(true);	Winner set to true	Winner set to true	
5	Check to see if get_winner returns the correct value	EXPECT_EQ(true, steven->get_winner());	True	True	Get_winner returns correct value of winner

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: OC 3 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests OPLCandidate setWinnerTest () return value.

Repo-Team13/Project1/testing/OPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: OPLCandidate compiles and a Candidate object is initialized correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	OPLCandidate jenna initialized	OPLCandidate* jenna = new OPLCandidate("Jenna America");			
3	Default get_winner value is checked	EXPECT_EQ(false, jenna->get_winner());	True	True	
4	Jenna set to winner as true	jenna->set_winner(true); EXPECT_EQ(true, jenna->get_winner());	True	True	
5	Jenna set to winner as false	jenna->set_winner(false); EXPECT_EQ(false, jenna->get_winner()); EXPECT_NE(true, jenna->get_winner());	True	True	

Post condition(s) for Test:

Set_winner correctly sets the winner status to the parameter that is entered.

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __

Test Date: 22/03/2024

Test Case ID#: OC 4

Name(s) of Testers: Michael Mulhall

Test Description: Tests OPLCandidate toStringTest() return value.

Repo-Team13/Project1/testing/OPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: OPLCandidate compiles and a Candidate object is initialized correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	OPLCandidate tom initialized	OPLCandidate* tom = new OPLCandidate("Tom Clancy");			
3	Tom's votes set to 5	Tom->set_num_votes(5);	Num_votes value set to 5	Num_votes value set to 5	
4	Expect to_string to output "5 Tom Clancy"	EXPECT_EQ("5 Tom Clancy", tom->to_string());	True	True	Using winner default value of 5
5	Tom set_winner(true)	Tom->set_winner(true);	Winner = true	Winner = true	
6	Tom to_string expected to return "5 Tom Clancy WINNER"	EXPECT_EQ("> 5 Tom Clancy - WINNER", tom->to_string()); EXPECT_NE("5 Tom Clancy", tom->to_string());	True	True	To_string returns the correct values

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: OC 5 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests OPLCandidate getandsetNumVotesTest() return value/ altered value.

Repo-Team13/Project1/testing/OPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: OPLCandidate compiles and a Candidate object is initialized correctly.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	OPLCandidate jerry initialized	OPLCandidate* jerry = new OPLCandidate("Jerry Seinfeld");	Jerry initialized	Jerry initialized	
3	Testing Jerry's expected default num_votes value of zero	EXPECT_EQ(0, jerry->get_num_votes());	True	True	Get_num_votes() returns correct default value of 0
4	Jerry's votes set to 6	jerry->set_num_votes(6);	Num_votes set to 5	Num_votes set to 5	Using winner default value of 5
5	Expect Jerry's votes to be 6	EXPECT_EQ(6, jerry->get_num_votes());	True	True	
6	Set Jerry's votes to new value 0	jerry->set_num_votes(0);	Num_votes set to 0	Num_votes set to 0	
7	Expect Jerry's votes to be 0	EXPECT_EQ(0, jerry->get_num_votes());	True	True	Set_votes correctly alters the num_votes value

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit X System __

Test Date: 22/03/2024

Test Case ID#: CC 1

Name(s) of Testers: Michael Mulhall

Test Description: Tests CPLCandidate get_name() return value.

Repo-Team13/Project1/testing/CPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: CPLCandidate compiles and a Candidate object is initialized correctly.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Create CPLCandidate object	CPLCandidate* joe = new CPLCandidate(“Joe Schmo”)	None	None	Candidate joe is initialized
3	Compare get_name to expected result	EXPECT_EQ(“Joe Schmo”, joe->get_name());	True	True	joe->get_name() returns Joe Schmo
4	Create CPLCandidate object	CPLCandidate* sam = new CPLCandidate(“Sam Politician”)	None	None	Candidate sam is initialized
5	Compare get_name to expected result	EXPECT_EQ(“Sam Politician”, sam->get_name());	True	True	sam->get_name() returns Sam Politician

Post condition(s) for Test:

Get_name correctly outputs a Candidate’s name. It can be used in the to_string function.

Project Name: Project 1: Voting System Team#13

Test Stage: Unit X System __

Test Date: 22/03/2024

Test Case ID#: CC 2

Name(s) of Testers: Michael Mulhall

Test Description: Tests CPLCandidate get_winner() return value.

repo-Team13/Project1/testing/CPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: CPLCandidate compiles and a CPLCandidate object is initialized correctly.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Candidate steven initialized	CPLCandidate* steven = new CPLCandidate ("Steven Carter");	Candidate initialized	Candidate initialzed	Candidate initialized
3	Compare winner variable with default winner value of false	EXPECT_EQ(false, steven->get_winner());	True	True	Candidate initialized with a winner value of false
4	Set winner to true	steven->set_winner(true);	Winner set to true	Winner set to true	
5	Check to see if get_winner returns the correct value	EXPECT_EQ(true, steven->get_winner());	True	True	Get_winner returns correct value of winner

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit X System __

Test Date: 22/03/2024

Test Case ID#: CC 3

Name(s) of Testers: Michael Mulhall

Test Description: Tests CPLCandidate set_winner() return value.

Repo-Team13/Project1/testing/CPLCandidateUnitTest.cpp

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: CPLCandidate compiles and a Candidate object is initialized correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	OPLCandidate jenna initialized	CPLCandidate* jenna = new CPLCandidate("Jenna America");			
3	Default get_winner value is checked	EXPECT_EQ(false, jenna->get_winner());	True	True	
4	Jenna set to winner as true	jenna->set_winner(true); EXPECT_EQ(true, jenna->get_winner());	True	True	
5	Jenna set to winner as false	jenna->set_winner(false); EXPECT_EQ(false, jenna->get_winner()); EXPECT_NE(true, jenna->get_winner());	True	True	

Post condition(s) for Test:

Set_winner correctly sets the winner status to the parameter that is entered.

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: CC 4 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests CPLCandidate toStringTest() return value.

Repo-Team13/Project1/testing/CPLCandidateUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: CPLCandidate compiles and a Candidate object is initialized correctly.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	CPLCandidate tom initialized	CPLCandidate* tom = new CPLCandidate("Tom Clancy");			
3	Tom's votes set to 5	Tom->set_num_votes(5);	Num_votes value set to 5	Num_votes value set to 5	
4	Expect to_string to output "5 Tom Clancy"	EXPECT_EQ("5 Tom Clancy", tom->to_string());	True	True	Using winner default value of 5
5	Tom set_winner(true)	Tom->set_winner(true);	Winner = true	Winner = true	
6	Tom to_string expected to return "5 Tom Clancy WINNER"	EXPECT_EQ("> 5 Tom Clancy - WINNER", tom->to_string()); EXPECT_NE("5 Tom Clancy", tom->to_string());	True	True	To_string returns the correct values

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: E1 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests ElectionData break_tie return value.

Repo-Team13/Project1/testing/ElectionDataUnitTest.cpp

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: ElectionData object is initialized. Break tie is between at least two candidates.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Integer num1 to be set equal to break_tie(5)	int num1 = ElectionData::break_tie(5);	Num1 should be a random integer in a range of 0 to 4		
3	Expect num1 to be in a range of 0 to 4	EXPECT_GE(num1, 0); EXPECT_LE(num1, 4);	True	True	
4	Expect num2 to be set equal to break_tie(72)	int num2 = ElectionData::break_tie(72);	Num2 should be a random integer between 0 and 71		
5	Expect num2 to be in a range of 0 to 71	EXPECT_GE(num2, 0); EXPECT_LE(num2, 71);	True	True	
6	Integer num3 to be set equal to break_tie(2)	int num3 = ElectionData::break_tie(2);	Num3 should be a random integer between 0 and 1 inclusive.		
	Expect num3 to be in a range of 0 to 1	EXPECT_GE(num3, 0); EXPECT_LE(num3, 1);	True	True	

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit X System __ **Test Date: 22/03/2024**

Test Case ID#: E2 **Name(s) of Testers: Michael Mulhall**

Test Description: Tests multiple ElectionData objects running consecutively.

Repo-Team13/Project1/testing/ElectionDataUnitTest.cpp

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: 3 ElectionData objects initialized.

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Initialize three ElectionData pointer objects elec1 to elec3.	ElectionData* elec1 = ElectionDataParser::create_election("../testing/test_data/1_person_cpl.csv"); ElectionData* elec2 = ElectionDataParser::create_election("../testing/test_data/2_party_opl.csv"); ElectionData* elec3 = ElectionDataParser::create_election("../testing/test_data/1_person_cpl.csv");	N/A	N/A	
3	elec1 is displayed and its output is compared to expected value	elec1->display(); output = testing::internal::GetCapturedStdout(); EXPECT_EQ(output, "2 Democratic:\n\t> Gary - WINNER\n");	True	True	
4	elec2 is displayed and its output is compared to expected value	testing::internal::CaptureStdout(); elec2->display(); output = testing::internal::GetCapturedStdout(); EXPECT_EQ(output, "6 Republican:\n\t> 4 Alawa - WINNER\n\t2 Etta\n3 Democrat:\n\t> 2 Pike - WINNER\n\t1 Lucy\n\t0 Beiye\n");	True	True	
5	elec3 is displayed and its output is compared to expected value	elec3->display(); output = testing::internal::GetCapturedStdout(); EXPECT_EQ(output, "2 Democratic:\n\t> Gary - WINNER\n");	True	True	

6	Initialize ElectionData pointer objects elec4 and elec5	ElectionData* elec4 = ElectionDataParser::create_election("../testing/test_data/1_person_cpl.csv"); ElectionData* elec5 = ElectionDataParser::create_election("../testing/test_data/2_party_opl.csv");	N/A	N/A	
7	elec4 is displayed and compared to expected value	elec4->display(); output = testing::internal::GetCapturedStdout(); EXPECT_EQ(output, "2 Democratic:\n\t> Gary - WINNER\n");	True	True	
8	elec5 is displayed and compared to expected value	elec5->display(); output = testing::internal::GetCapturedStdout(); EXPECT_EQ(output, "6 Republican:\n\t> 4 Alawa - WINNER\n\t2 Etta\n3 Democrat:\n\t> 2 Pike - WINNER\n\t1 Lucy\n\t0 Beiye\n");	True	True	No memory issues have been detected through valgrind when running multiple elections at the same time.

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __ **Test Date: March 22nd**

Test Case ID#: AL1 **Name(s) of Testers: Khalid Qasim**

Test Description:

This test verifies the Audit Log constructor initializes the log with an empty string

Filename: AuditLogUnitTest.cpp
Testname: AuditLogTest, ConstructorTest
Functions: AuditLog()

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call write_to_file method without adding any content to the log	Filename = "ConstructorTest.txt"	N/A	N/A	
2	Construct a path to the file	testDirectory and Filename = "ConstructorTest.txt"	N/A	N/A	
3	Read the file content to a string	Path to "ConstructorTest.txt"	N/A	N/A	
4	Compare file results with expected results	"ConstructorTest.txt" file content	"" (empty string)	"" (empty string)	

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __

Test Date: March 22nd

Test Case ID#: AL2

Name(s) of Testers: Khalid Qasim

Test Description:

This test verifies if the add_line method correctly adds a line to the log

Filename: AuditLogUnitTest.cpp
Testname: AuditLogTest, AddLineTest
Functions: add_line()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call add_line method with testLine	testLine = "Test line for AddLineTest";	N/A	N/A	
2	Write the log content to a file	Filename = "AddLineTest.txt"	N/A	N/A	
3	Construct a path to the file	testDirectory and Filename = "AddLineTest.txt"	N/A	N/A	
4	Write auditlog to "AddLineTest.txt"	auditLog.write_to_file(testDirectory, filename);	N/A	N/A	
5	Read the file content to a string	Path to "AddLineTest.txt"	N/A	N/A	
6	Compare file results with expected results	"AddLineTest.txt" file content	"Test line for AddLineTest\n"	"Test line for AddLineTest\n"	

Post condition(s) for Test:

Project Name: Project 1: Voting System Team# 13

Test Stage: Unit _x_ System __

Test Date: March 22nd

Test Case ID#: AL3

Name(s) of Testers: Khalid Qasim

Test Description:

This test verifies if the clear_log method correctly clears the logs content to an empty string

Filename: AuditLogUnitTest.cpp
Testname: AuditLogTest, ClearLogTest
Functions: clear_log()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call clear_log method with the test line	“Test line for ClearLogTest”	N/A	N/A	
2	Write the log content to a file	Filename = “ClearLogTest.txt”	N/A	N/A	
3	Construct a path to the file	testDirectory and Filename = “ClearLogTest.txt”	N/A	N/A	
4	Read the file content to a string	Path to “ClearLogTest.txt”	N/A	N/A	
5	Compare file results with expected results	“ClearLogTest.txt” file content	“” (no content)	“” (no content)	Verify no content in the file

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __

Test Date: March 22nd

Test Case ID#: AL4

Name(s) of Testers: Khalid Qasim

Test Description:

This test verifies if the add_allocation_table method correctly formats and adds the allocation table to the log for a CPL election with a tie, this is a visual test

Filename: main.cpp
Testname: NA
Functions: add_allocation_table()

Automated: yes no X

Results: Pass __X___ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the program from main on 3_person_tie_cpl.csv and add to the AuditLog				

2	select ‘y’ to generate audit file				
3	Review the data written to audit file against expected outputs for match		True	True	

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __ **Test Date:** **March 22nd**

Test Case ID#: AL5 **Name(s) of Testers: Khalid Qasim**

Test Description:

This test verifies if the add_allocation_table method correctly formats and adds the allocation table to the log for an OPL election with ties, this is a visual test

Filename: main.cpp
Testname: NA
Functions: add_allocation_table()

Automated: yes no X

Results: Pass __X__ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the program from main on sys_test2_opl.csv and add to the AuditLog				
2	select ‘y’ to generate audit file				
3	Review the data written to audit file against expected outputs for match		True	True	
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __ **Test Date: March 22nd**

Test Case ID#: AL6 **Name(s) of Testers: Khalid Qasim**

Test Description:

This test verifies if the tie breaker data is properly sent to the audit log for a CPL election, this is a visual test

Filename: main.cpp
Testname: NA
Functions: add_line(), calculate_seats_per_party()

Automated: yes___ no _X__

Results: Pass __X__ Fail________

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run the program from main on 3_person_tie_cpl.csv and add to the AuditLog				
2	select ‘y’ to generate audit file				
3	Review the data written to audit file against expected outputs for match		True	True	
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __ **Test Date: March 22nd**

Test Case ID#: AL7 **Name(s) of Testers: Khalid Qasim**

Test Description:

This test verifies if the tie breaker data is properly sent to the audit log for a OPL election, this is a visual test

Filename: main.cpp
Testname: NA
Functions: add_line(), calculate_seats_per_party()

Automated: yes___ no _X__

Results: Pass Fail X

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the program from main on sys_test2_opl.csv and add to the AuditLog				
2	select 'y' to generate audit file				
3	Review the data written to audit file against expected outputs for match		True	False	See buglist, we are currently unable to send our opl tie-breaker data to the auditlog
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __

Test Date: March 22nd

Test Case ID#: AL8

Name(s) of Testers: Khalid Qasim

Test Description:
This test verifies if the write_to_file method correctly write the current log to a specified file

Filename: AuditLogUnitTest.cpp
Testname: AuditLogTest, WriteToFileTest
Functions: write_to_file()

Automated: yes X no

Results: Pass __X___ Fail_________

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call add_line method with testLine	testLine = "Test line for WriteToFileTest";	N/A	N/A	
2	Write the log content to a file	Filename = "WriteToFileTest.txt"	N/A	N/A	
3	Construct a path to the file	testDirectory and Filename = "WriteToFileTest.txt"	N/A	N/A	
4	Read the file content to a string	Path to "WriteToFileTest.txt"	N/A	N/A	
5	Compare file results with expected results	"WriteToFileTest.txt" file content	testLine + "\n"	testLine + "\n"	

Post condition(s) for Test:

Project Name: Project 1: Voting System Team# 13

Test Stage: Unit _x_ System __

Test Date: April 8th

Test Case ID#: AL9

Name(s) of Testers: Michael Mulhall

Test Description:
This test verifies if the write_to_file method correctly write the current log to a specified file

Filename: AuditLogUnitTest.cpp
Testname: AuditLogTest, MultipleFilesSameSecondTest
Functions: write_to_file()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call add_line method with testLine	testLine = "Test line for WriteToFileTest";	N/A	N/A	
2	Write the log content to 3 different files consecutively to ensure at least two are in the same second.	auditLog.write_to_file() auditLog.write_to_file() auditLog.write_to_file()	N/A	N/A	

3	Check how many audit log files were created	glob_t gl;
		size_t num = 0;
		if(glob("audit_*", GLOB_NOSORT, NULL, &gl) == 0)
		    num = gl.gl_pathc;
		globfree(&gl);
			num = 3	num = 3	

4	Check if num is equal to 3	EXPECT_EQ(num, 3)			Bug fixed. Multiple audit logs can be generated in the same second.
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System   Team# 13

Test Stage: Unit _X_   System __

Test Date: April 14th

Test Case ID#: AL10

Name(s) of Testers: Michael Mulhall

Test Description:
This test verifies there are no additional commas on the end of the break_tie array in the audit log.

Filename: ElectionData.cpp
Testname: VisualTieBreakerArray
Functions: calculate_seats_per_party

Automated: yes   no   X

Results: Pass __X___   Fail________

Preconditions for Test: Audit Log object instantiated and test directory exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run an election with test file 2_people_cpl_tie.csv	make; make run 2_people_cpl_tie.csv y			
2	Visually check to ensure audit log break_tie array does not have a comma after the last element	N/A	True	True	visually checked
3	run an election with test file 3_person_tie_cpl.csv	make; make run 3_person_cpl_tie.csv y			
4	Visually check to ensure audit log break_tie array does not have a comma after the last element				visually checked
5					

Post condition(s) for Test:

Project Name: Voting System

Team #13

Test Stage: Unit _X_ System __

Test Date: 3/26/24

Test Case ID#: OPL_ED_Display1

Name(s) of Testers: Connell Hagen

Test Description:

Tests the display() function

setup with:

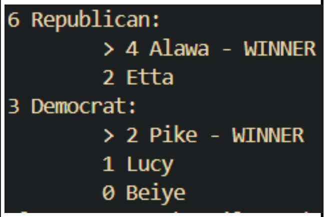
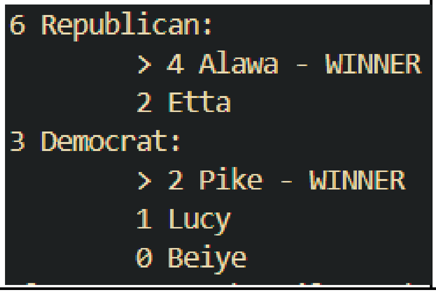
```
ElectionData* std_case_1 =
ElectionDataParser::create_election("testing/test_data/2_party_opl.csv");
std_case_1->display();
```

Filename: OPLElectionDataUnitTest.cpp
Testname: OPLElectionDataUnitTest, Display
Functions: display()

Automated: yes _X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up an OPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OPLElectionData Object	testing/test_data/2_party_opl.csv	std_case_1 != nullptr	std_case_1 != nullptr	
2	Display Election results	terminal output	 6 Republican: > 4 Alawa - WINNER 2 Etta 3 Democrat: > 2 Pike - WINNER 1 Lucy 0 Beiye	 6 Republican: > 4 Alawa - WINNER 2 Etta 3 Democrat: > 2 Pike - WINNER 1 Lucy 0 Beiye	

Post condition(s) for Test:

`display()` outputs all `to_string()` representations of its aggregated `Party`s to the terminal. All `Candidate`s within the same party are also sorted by number of votes received.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: OPL_ED_Display2 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the display() function's performance

setup with:

```
clock_t t = clock();
ElectionData* opl_100000 =
ElectionDataParser::create_election("testing/test_data/100000_votes_opl.csv");
opl_100000->display();
const double work_time = (clock() - t) / double(CLOCKS_PER_SEC);
```

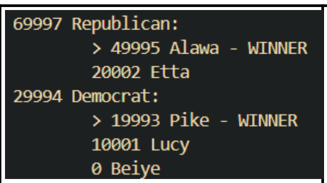
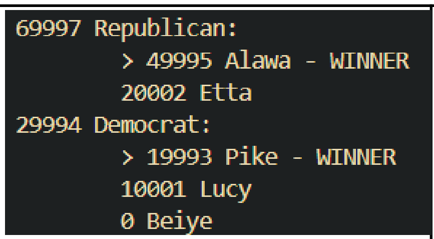
Filename: OPLElectionDataUnitTest.cpp
Testname: OPLElectionDataUnitTest, Display
Functions: display()

Automated: yes_X_ no ___

Results: Pass ___X_ Fail________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up an OPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Start Timer				
2	Create CPLElectionData Object	testing/test_data/100000_votes_opl.csv	opl_100000 != nullptr	opl_100000 != nullptr	The data will be corrupt if an election is created before this current election is, as it is in the test file. This is a bug currently in the buglist.

3	Display Election results	terminal output	```
69997 Republican:
> 49995 Alawa - WINNER
20002 Etta
29994 Democrat:
> 19993 Pike - WINNER
10001 Lucy
0 Beiye
```	```
69997 Republican:
> 49995 Alawa - WINNER
20002 Etta
29994 Democrat:
> 19993 Pike - WINNER
10001 Lucy
0 Beiye
``	
4	Check Timer	work_time	work_time <= 4 * 60	work_time <= 4 * 60	

Post condition(s) for Test:

\`display()\` outputs all \`to_string()\` representations of its aggregated \`Party\`s to the terminal. A 100,000 vote election can be completely tabulated within 4 minutes.
```

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: OPL_ED_AuditLog1 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the generate_audit_log() function

setup with:
ElectionData* test =
ElectionDataParser::create_election(“testing/test_data/2_party_opl.csv”);

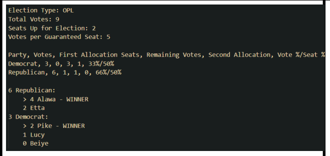
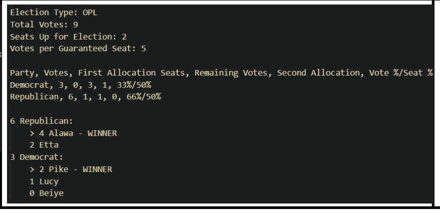
Filename: n/a
Testname: OPLElectionDataUnitTest, AuditLog
Functions: generate_audit_file()

Automated: yes__ no _X__

Results: Pass X Fail

Preconditions for Test: The ElectionDataParser create_election() function properly sets up an OPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OPLElectionData Object	testing/test_data/2_party_opl.csv	test != nullptr	test != nullptr	

2	Generate Audit File	file output	 `Election Type: OPL`
			`Total Votes: 9`
			`Seats Up for Election: 2`
			`Votes per Guaranteed Seat: 5`
			
			`Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %`
			`Democrat, 3, 0, 3, 1, 33%/50%`
			`Republican, 6, 1, 1, 0, 66%/50%`
			
			`6 Republican:`
			`    > 4 Alawa - WINNER`
			`    2 Etta`
			`3 Democrat:`
			`    > 2 Pike - WINNER`
			`    1 Lucy`
			`    0 Beiye`
				 `Election Type: OPL`
				`Total Votes: 9`
				`Seats Up for Election: 2`
				`Votes per Guaranteed Seat: 5`
				
				`Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %`
				`Democrat, 3, 0, 3, 1, 33%/50%`
				`Republican, 6, 1, 1, 0, 66%/50%`
				
				`6 Republican:`
				`    > 4 Alawa - WINNER`
				`    2 Etta`
				`3 Democrat:`
				`    > 2 Pike - WINNER`
				`    1 Lucy`
				`    0 Beiye`	

Post condition(s) for Test:

`generate_audit_file()` outputs a header including the type of election and basic statistics relating to it, all ties that were broken during the calculations, a table with a breakdown of the calculation of seat awarding, and all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: OPL_ED_AuditLog2 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the generate_audit_log() function

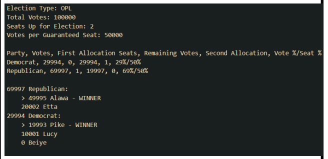
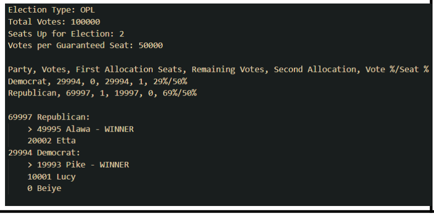
setup with:
ElectionData* test =
ElectionDataParser::create_election("testing/test_data/100000_opl.csv");

Filename: n/a
Testname: OPLElectionDataUnitTest, AuditLog
Functions: generate_audit_file()

Automated: yes__ no _X__

Results: Pass__ X_ Fail

Preconditions for Test: The ElectionDataParser create_election() function properly sets up an OPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OPLElectionData Object	testing/test_data/100000_opl.csv	test != nullptr	test != nullptr	
2	Generate Audit File	file output	 Election Type: OPL Total Votes: 100000 Seats Up for Election: 2 Votes per Guaranteed Seat: 50000 Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat % Democrat, 29994, 0, 29994, 1, 29%/50% Republican, 69997, 1, 19997, 0, 69%/50% 69997 Republican: > 49995 Alawa - WINNER 20002 Etta 29994 Democrat: > 19993 Pike - WINNER 10001 Lucy 0 Beiye	 Election Type: OPL Total Votes: 100000 Seats Up for Election: 2 Votes per Guaranteed Seat: 50000 Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat % Democrat, 29994, 0, 29994, 1, 29%/50% Republican, 69997, 1, 19997, 0, 69%/50% 69997 Republican: > 49995 Alawa - WINNER 20002 Etta 29994 Democrat: > 19993 Pike - WINNER 10001 Lucy 0 Beiye	

Post condition(s) for Test:

`generate_audit_file()` outputs a header including the type of election and basic statistics relating to it, all ties that were broken during the calculations, a table with a breakdown of the calculation of seat awarding, and all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit __ System _X_ **Test Date:** 4/15/24

Test Case ID#: OPL_ED_Mult_1 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the OPLElectionData with multiple files

setup with:
ElectionData* opl1 =
ElectionDataParser::create_election(std::vector<std::string>({"../testing/test_data/opl_mult1_1.csv","../testing/test_data/opl_mult1_2.csv","../testing/test_data/opl_mult1_3.csv","../testing/test_data/opl_mult1_4.csv"}));;

Filename: n/a
Testname: OPLElectionDataUnitTest, OPLParsing Strategy
Functions: create_election()

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: The ElectionDataParser create_election() function is passed valid and accessible file paths

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OPLElectionData Object	opl1 = ElectionDataParser::create_el			

		ection(std::vector<std::string>({"../testing/test_data/opl_mult1_1.csv","../testing/test_data/opl_mult1_2.csv","../testing/test_data/opl_mult1_3.csv","../testing/test_data/opl_mult1_4.csv"}));			
2	Compare terminal output (GetCapturedStdout) with expected string display	terminal output	24 Republican: > 13 Etta - WINNER 11 Alawa 17 Democrat: > 9 Pike - WINNER 8 Lucy 0 Beiye	24 Republican: > 13 Etta - WINNER 11 Alawa 17 Democrat: > 9 Pike - WINNER 8 Lucy 0 Beiye	

Post condition(s) for Test:

Project Name: Voting System **Team #13**

Test Stage: Unit __ System _X_ **Test Date: 4/15/24**

Test Case ID#: OPL_ED_Mult_2 **Name(s) of Testers: Grant Oie**

Test Description:
Tests the OPLElectionData with multiple files

setup with:
ElectionData* opl1 =
ElectionDataParser::create_election(std::vector<std::string>({"../testing/test_data/opl_mult2_1.csv","../testing/test_data/opl_mult2_2.csv"}));;

Filename: n/a
Testname: OPLElectionDataUnitTest, OPLParsing Strategy
Functions: create_election()

Automated: yes _X_ no ___

Results: Pass X Fail

Preconditions for Test: The ElectionDataParser create_election() function is passed valid and accessible file paths

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OPLElectionData Object	opl1 = ElectionDataParser::create_election(std::vector<std::string>({"../testing/test_data/opl_mult2_1.csv","../testing/test_data/opl_mult2_2.csv"}));			
2	Compare terminal output (GetCapturedStdout) with expected string display	terminal output	7 Democratic: > 7 Gary - WINNER	7 Democratic: > 7 Gary - WINNER	

Post condition(s) for Test:

Project Name: Voting System

Team #13

Test Stage: Unit _X_ System __

Test Date: 3/26/24

Test Case ID#: CPL_ED_Display1

Name(s) of Testers: Connell Hagen

Test Description:

Tests the display() function

setup with:

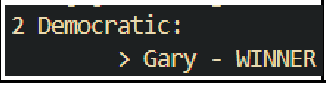
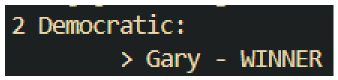
```
ElectionData* cpl_1p =
ElectionDataParser::create_election("testing/test_data/1_person_cpl.csv");
cpl_1p->display();
```

Filename: CPLElectionDataUnitTest.cpp
Testname: CPLElectionDataUnitTest, Display
Functions: display()

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create CPLElectionData Object	testing/test_data/1_person_cpl.csv	cpl_1p != nullptr	cpl_1p != nullptr	
2	Display Election results	terminal output	 `2 Democratic:` `> Gary - WINNER`	 `2 Democratic:` `> Gary - WINNER`	

Post condition(s) for Test:

`display()` outputs all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: CPL_ED_Display2 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the display() function

setup with:
ElectionData* std_case_1 =
ElectionDataParser::create_election("testing/test_data/sys_test3_cpl.csv");
std_case_1->display();

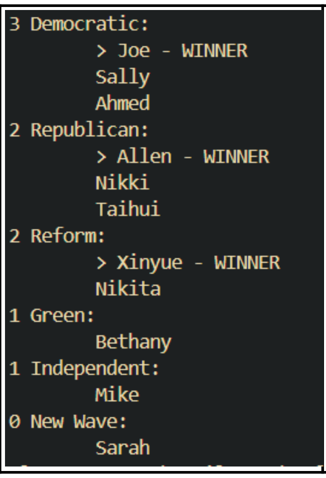
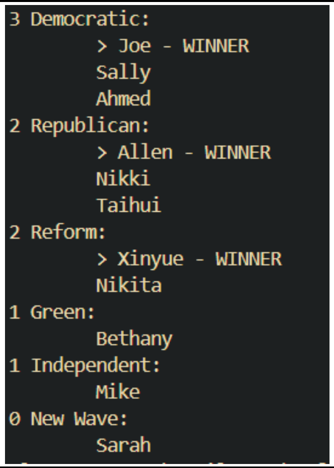
Filename: CPLElectionDataUnitTest.cpp
Testname: CPLElectionDataUnitTest, Display
Functions: display()

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create CPLElectionData Object	testing/test_data/sys_test3_cpl.csv	std_case_1 != nullptr	std_case_1 != nullptr	

2	Display Election results	terminal output			

```
3 Democratic:
        > Joe - WINNER
        Sally
        Ahmed
2 Republican:
        > Allen - WINNER
        Nikki
        Taihui
2 Reform:
        > Xinyue - WINNER
        Nikita
1 Green:
        Bethany
1 Independent:
        Mike
0 New Wave:
        Sarah
```

```
3 Democratic:
        > Joe - WINNER
        Sally
        Ahmed
2 Republican:
        > Allen - WINNER
        Nikki
        Taihui
2 Reform:
        > Xinyue - WINNER
        Nikita
1 Green:
        Bethany
1 Independent:
        Mike
0 New Wave:
        Sarah
```

Post condition(s) for Test:

`display()` outputs all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: CPL_ED_Display3 **Name(s) of Testers: Connell Hagen**

Test Description:

Tests the display() function's performance

setup with:

```
clock_t t = clock();
ElectionData* cpl_100000 =
ElectionDataParser::create_election("testing/test_data/100000_votes_cpl.csv");
cpl_100000->display();
const double work_time = (clock() - t) / double(CLOCKS_PER_SEC);
```

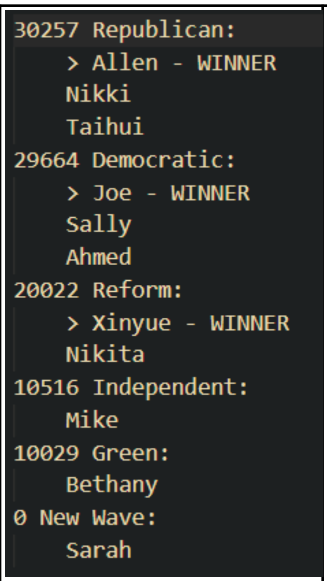
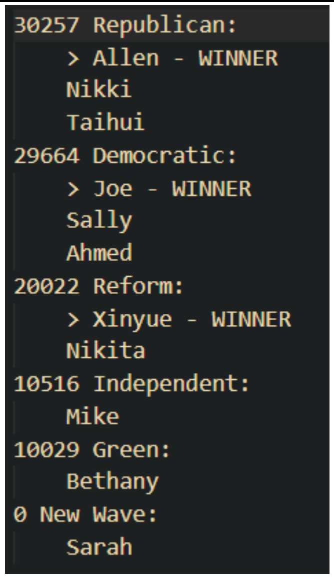
Filename: CPLElectionDataUnitTest.cpp
Testname: CPLElectionDataUnitTest, Display
Functions: display()

Automated: yes_X_ no ___

Results: Pass ___X_ Fail_________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Start Timer				
2	Create CPLElectionData Object	testing/test_data/100000_votes_cpl.csv	cpl_100000 != nullptr	cpl_100000 != nullptr	

3	Display Election results	terminal output			
4	Check Timer	work_time	work_time <= 4 * 60	work_time <= 4 * 60	

```
30257 Republican:
    > Allen - WINNER
    Nikki
    Taihui
29664 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
20022 Reform:
    > Xinyue - WINNER
    Nikita
10516 Independent:
    Mike
10029 Green:
    Bethany
0 New Wave:
    Sarah
```

```
30257 Republican:
    > Allen - WINNER
    Nikki
    Taihui
29664 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
20022 Reform:
    > Xinyue - WINNER
    Nikita
10516 Independent:
    Mike
10029 Green:
    Bethany
0 New Wave:
    Sarah
```

Post condition(s) for Test:

`display()` outputs all `to_string()` representations of its aggregated `Party`s to the terminal. A 100,000 vote election can be completely tabulated within 4 minutes.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: CPL_ED_AuditLog1 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the generate_audit_log() function

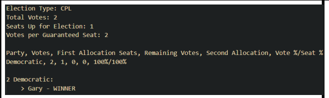
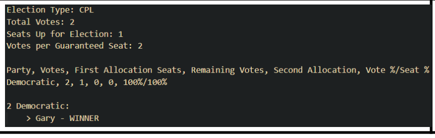
setup with:
ElectionData* test =
ElectionDataParser::create_election("testing/test_data/1_person_cpl.csv");

Filename: n/a
Testname: CPLElectionDataUnitTest, AuditLog
Functions: generate_audit_file()

Automated: yes__ no _X__

Results: Pass X Fail

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create CPLElectionData Object	testing/test_data/1_person_cpl.csv	test != nullptr	test != nullptr	
2	Generate Audit File	file output			

Expected Result (step 2):

```
Election Type: CPL
Total Votes: 2
Seats Up for Election: 1
Votes per Guaranteed Seat: 2

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 2, 1, 0, 0, 100%/100%

2 Democratic:
    > Gary - WINNER
```

Actual Result (step 2):

```
Election Type: CPL
Total Votes: 2
Seats Up for Election: 1
Votes per Guaranteed Seat: 2

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 2, 1, 0, 0, 100%/100%

2 Democratic:
    > Gary - WINNER
```

Post condition(s) for Test:

`generate_audit_file()` outputs a header including the type of election and basic statistics relating to it, all ties that were broken during the calculations, a table with a breakdown of the calculation of seat awarding, and all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __ **Test Date: 3/26/24**

Test Case ID#: CPL_ED_AuditLog2 **Name(s) of Testers: Connell Hagen**

Test Description:
Tests the generate_audit_log() function

setup with:
ElectionData* test =
ElectionDataParser::create_election("testing/test_data/sys_test3_cpl.csv");

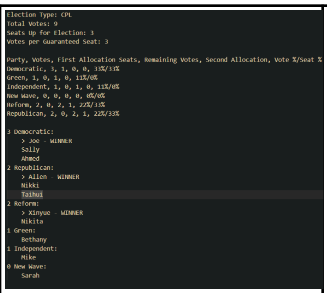
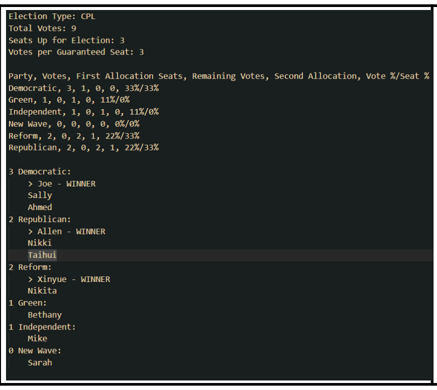
Filename: n/a
Testname: CPLElectionDataUnitTest, AuditLog
Functions: generate_audit_file()

Automated: yes__ no _X__

Results: Pass X Fail

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create CPLElectionData Object	testing/test_data/sys_test3_cpl.csv	test != nullptr	test != nullptr	

2	Generate Audit File	file output			

```
Election Type: CPL
Total Votes: 9
Seats Up for Election: 3
Votes per Guaranteed Seat: 3

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 3, 1, 0, 0, 33%/33%
Green, 1, 0, 1, 0, 11%/0%
Independent, 1, 0, 1, 0, 11%/0%
New Wave, 0, 0, 0, 0, 0%/0%
Reform, 2, 0, 2, 1, 22%/33%
Republican, 2, 0, 2, 1, 22%/33%

3 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
2 Republican:
    > Allen - WINNER
    Nikki
    Taihui
2 Reform:
    > Xinyue - WINNER
    Nikita
1 Green:
    Bethany
1 Independent:
    Mike
0 New Wave:
    Sarah
```

```
Election Type: CPL
Total Votes: 9
Seats Up for Election: 3
Votes per Guaranteed Seat: 3

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 3, 1, 0, 0, 33%/33%
Green, 1, 0, 1, 0, 11%/0%
Independent, 1, 0, 1, 0, 11%/0%
New Wave, 0, 0, 0, 0, 0%/0%
Reform, 2, 0, 2, 1, 22%/33%
Republican, 2, 0, 2, 1, 22%/33%

3 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
2 Republican:
    > Allen - WINNER
    Nikki
    Taihui
2 Reform:
    > Xinyue - WINNER
    Nikita
1 Green:
    Bethany
1 Independent:
    Mike
0 New Wave:
    Sarah
```

Post condition(s) for Test:

`generate_audit_file()` outputs a header including the type of election and basic statistics relating to it, all ties that were broken during the calculations, a table with a breakdown of the calculation of seat awarding, and all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Voting System **Team #13**

Test Stage: Unit _X_ System __

Test Date: 3/26/24

Test Case ID#: CPL_ED_AuditLog3

Name(s) of Testers: Connell Hagen

Test Description:
Tests the generate_audit_log() function

setup with:
ElectionData* test =
ElectionDataParser::create_election("testing/test_data/100000_votes_cpl.csv.csv"
);

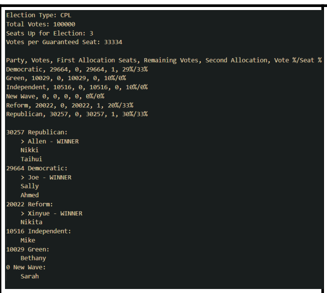
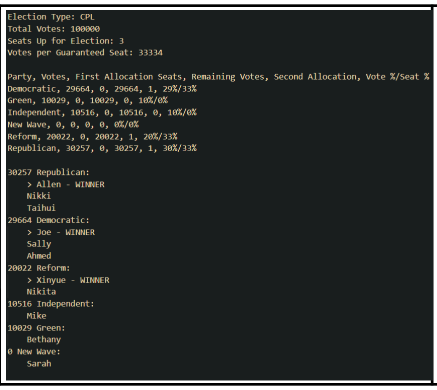
Filename: n/a
Testname: CPLElectionDataUnitTest, AuditLog
Functions: generate_audit_file()

Automated: yes__ no _X__

Results: Pass ___X_ Fail________

Preconditions for Test: The ElectionDataParser create_election() function properly sets up a CPL election according to all of its pre- and post-conditions. The winners of the election were properly calculated, and set within the Candidate objects.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create CPLElectionData Object	testing/test_data/100000_votes_cpl.csv.csv	test != nullptr	test != nullptr	

2	Generate Audit File	file output			

```
Election Type: CPL
Total Votes: 100000
Seats Up for Election: 3
Votes per Guaranteed Seat: 33334

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 29664, 0, 29664, 1, 29%/33%
Green, 10029, 0, 10029, 0, 10%/0%
Independent, 10516, 0, 10516, 0, 10%/0%
New Wave, 0, 0, 0, 0, 0%/0%
Reform, 20022, 0, 20022, 1, 20%/33%
Republican, 30257, 0, 30257, 1, 30%/33%

30257 Republican:
    > Allen - WINNER
    Nikki
    Taihui
29664 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
20022 Reform:
    > Xinyue - WINNER
    Nikita
10516 Independent:
    Mike
10029 Green:
    Bethany
0 New Wave:
    Sarah
```

```
Election Type: CPL
Total Votes: 100000
Seats Up for Election: 3
Votes per Guaranteed Seat: 33334

Party, Votes, First Allocation Seats, Remaining Votes, Second Allocation, Vote %/Seat %
Democratic, 29664, 0, 29664, 1, 29%/33%
Green, 10029, 0, 10029, 0, 10%/0%
Independent, 10516, 0, 10516, 0, 10%/0%
New Wave, 0, 0, 0, 0, 0%/0%
Reform, 20022, 0, 20022, 1, 20%/33%
Republican, 30257, 0, 30257, 1, 30%/33%

30257 Republican:
    > Allen - WINNER
    Nikki
    Taihui
29664 Democratic:
    > Joe - WINNER
    Sally
    Ahmed
20022 Reform:
    > Xinyue - WINNER
    Nikita
10516 Independent:
    Mike
10029 Green:
    Bethany
0 New Wave:
    Sarah
```

Post condition(s) for Test:

`generate_audit_file()` outputs a header including the type of election and basic statistics relating to it, all ties that were broken during the calculations, a table with a breakdown of the calculation of seat awarding, and all `to_string()` representations of its aggregated `Party`s to the terminal.

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 1

Test Description: Ballot file with 1 person for a CPL election.

repo-Team13/Project1/testing/testing_data/1_person_cpl.csv

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("1_person_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 2

Test Description: Ballot file with 1 person for a OPL election.

repo-Team13/Project1/testing/testing_data/1_person_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("1_person_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 3

Test Description: Ballot file with 2 parties and multiple candidates for an OPL Election

repo-Team13/Project1/testing/testing_data/2_party_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("2_party_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit __ System X

Test Date: **22/03/2024**

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 4

Test Description: Ballot file with 2 people, one in each party, and they tied.

repo-Team13/Project1/testing/testing_data/2_people_cpl_tie.csv

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: None

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

2	Election is created with correct file name	Election _Data e = Create_Election("2_people_cpl_tie.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 5

Test Description: Ballot file with 3 people, all in different parties, and they tied

repo-Team13/Project1/testing/testing_data/3_person_tie_cpl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Election is created with correct file name	Election _Data e = Create_Election("3_person_tie_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					

5					

Post condition(s) for Test:

Project Name: Project 1: Voting System   Team#13

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 6

Test Description: Ballot file with 100,000 votes for a CPL Election.

repo-Team13/Project1/testing/testing_data/100000_votes_cpl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Election is created with correct file name	Election _Data e = Create_Election("100000_votes_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 7

Test Description: Ballot file with 100,000 votes for a OPL Election.

repo-Team13/Project1/testing/testing_data/100000_votes_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("100000_votes_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 8

Test Description: Ballot file with 3 parties, 6 candidates, and no ties

repo-Team13/Project1/testing/testing_data/sys_test1_opl.csv

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test1_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 9

Test Description: Ballot file with 3 parties, 8 candidates, and no ties

repo-Team13/Project1/testing/testing_data/sys_test2_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test2_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 10

Test Description: Ballot file with 6 parties, 9 votes, and a quota of 3.

repo-Team13/Project1/testing/testing_data/sys_test3_cpl.csv

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: None

Step	**Test Step**	**Test**	**Expected**	**Actual**	

# 1	Description	Data	Result	Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test3_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 11

Test Description: Ballot file with 4 candidates, 4 parties, and no ties

repo-Team13/Project1/testing/testing_data/sys_test4_cpl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test4_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	

4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 12

Test Description: Ballot file with a quota of 3, 2 parties, 13 votes, and all but 1 vote go to one party.

repo-Team13/Project1/testing/testing_data/sys_test5_overload_cpl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test5_overload_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 13

Test Description: Ballot file with 4 candidates, 4 parties, and tying for OPL

repo-Team13/Project1/testing/testing_data/sys_test6_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test6_overloadseats_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 14

Test Description: Ballot file with a quota of 4, 8 votes, and 3 parties where every vote goes to the same party for CPL

repo-Team13/Project1/testing/testing_data/sys_test7_votesforsingleparty_cpl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test7_votesforsingleparty_cpl.csv")			
3	Election results are displayed into the terminal	e.display()	CPL Display test passes	CPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit __ System X

Test Date: 22/03/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: S 15

Test Description: Ballot file with a quota of 4, 8 votes, and 3 parties where every vote goes to the same party for OPL

repo-Team13/Project1/testing/testing_data/sys_test8_votesforsinglecandidate_opl.csv

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test8_votesforsinglecandidate_opl.csv")			
3	Election results are displayed into the terminal	e.display()	OPL Display test passes	OPL Display test passes	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit _X_ System

Test Date: 12/04/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: OPL_AL1

Test Description: testing bug fix, "See results of OPL Election Tie in Audit log"

repo-Team13/Project1/testing/testing_data/sys_test9_opl_tie.csv

Automated: yes no X_

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Election is created with correct file name	Election _Data e = Create_Election("sys_test9_opl_tie.csv")			
3	Audit log created and checked for accurate tie information	audit log	audit log tie breaker info accurate and present	audit log tie breaker info accurate and present	
4					
5					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team#13

Test Stage: Unit _X_ System X

Test Date: 04/15/2024

Test Case ID#: CPLElectionData_MultipleFiles_1

Name(s) of Testers: Connell Hagan

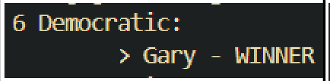
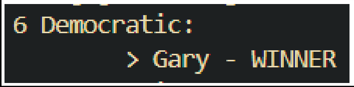
Test Description: Multiple CPL files can be aggregated into 1 election that is calculated.

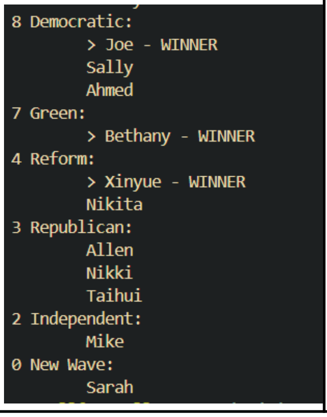
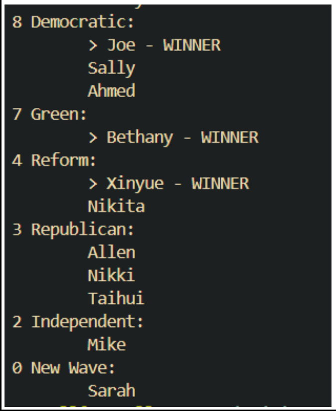
Filename: CPLElectionDataUnitTest.cpp
Testname: CPLElectionDataUnitTest, MultipleFiles
Functions: display()
repo-Team13/Project1/testing/testing_data/cpl_mult1_1.csv
repo-Team13/Project1/testing/testing_data/cpl_mult1_2.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_1.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_2.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_3.csv

Automated: yes X no _

Results: Pass X Fail

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create CPLElectionData Object	testing/testing_data/cpl_mult1_1.csv testing/testing_data/cpl_mult1_2.csv			
2	Create CPL ElectionData Object	testing/testing_data/cpl_mult2_1.csv testing/testing_data/cpl_mult2_2.csv testing/testing_data/cpl_mult2_3.csv			
3	Display	object1->display()	 `6 Democratic:` `        > Gary - WINNER`	 `6 Democratic:` `        > Gary - WINNER`	

4	Display	object2->display()	 `8 Democratic:`
			`        > Joe - WINNER`
			`        Sally`
			`        Ahmed`
			`7 Green:`
			`        > Bethany - WINNER`
			`4 Reform:`
			`        > Xinyue - WINNER`
			`        Nikita`
			`3 Republican:`
			`        Allen`
			`        Nikki`
			`        Taihui`
			`2 Independent:`
			`        Mike`
			`0 New Wave:`
			`        Sarah`
				 `8 Democratic:`
				`        > Joe - WINNER`
				`        Sally`
				`        Ahmed`
				`7 Green:`
				`        > Bethany - WINNER`
				`4 Reform:`
				`        > Xinyue - WINNER`
				`        Nikita`
				`3 Republican:`
				`        Allen`
				`        Nikki`
				`        Taihui`
				`2 Independent:`
				`        Mike`
				`0 New Wave:`
				`        Sarah`	

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team#13**

Test Stage: Unit _X_ System X **Test Date: 04/15/2024**

Test Case ID#: CPLElectionData_MultipleFiles_2

Name(s) of Testers: Connell Hagan

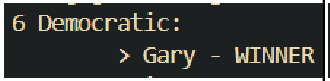
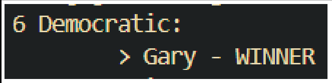
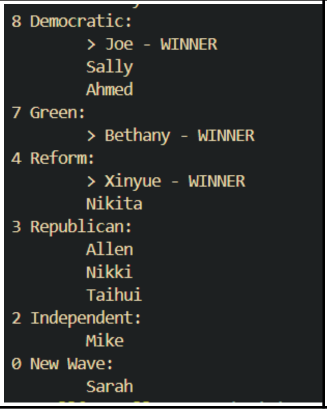
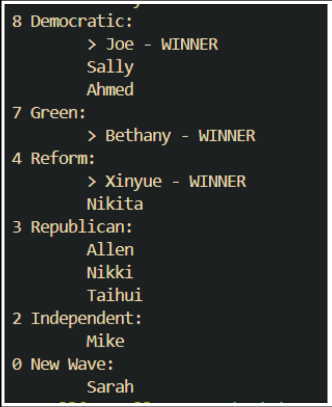
Test Description: Testing CPL Multiple File Functionality on the command line

repo-Team13/Project1/testing/testing_data/cpl_mult1_1.csv
repo-Team13/Project1/testing/testing_data/cpl_mult1_2.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_1.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_2.csv
repo-Team13/Project1/testing/testing_data/cpl_mult2_3.csv

Automated: yes no X

Results: Pass X Fail

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Run test 1	./election testing/test_data/cpl_mult1_1.csv testing/test_data/cpl_mult1_2.csv	 6 Democratic: > Gary - WINNER	 6 Democratic: > Gary - WINNER	
3	Run test 2	./election testing/testing_data/cpl_mult2_1.csv testing/testing_data/cpl_mult2_2.csv testing/testing_data/cpl_mult2_3.csv	 8 Democratic: > Joe - WINNER Sally Ahmed 7 Green: > Bethany - WINNER 4 Reform: > Xinyue - WINNER Nikita 3 Republican: Allen Nikki Taihui 2 Independent: Mike 0 New Wave: Sarah	 8 Democratic: > Joe - WINNER Sally Ahmed 7 Green: > Bethany - WINNER 4 Reform: > Xinyue - WINNER Nikita 3 Republican: Allen Nikki Taihui 2 Independent: Mike 0 New Wave: Sarah	

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/20/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MPO_MultipleFiles_1

Test Description: Multiple MPO files can be aggregated into 1 election that is calculated.

repo-Team13/Project2/testing/testing_data/mpo_mult1_1.csv
repo-Team13/Project2/testing/testing_data/mpo_mult1_2.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_1.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_2.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_3.csv

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
				N/A	
1	Create MPOElectionData Object	testing/testing_data/mpo_mult1_1.csv testing/testing_data/mpo_mult1_2.csv			
2	Create MPOElectionData Object	testing/testing_data/mpo_mult2_1.csv testing/testing_data/mpo_mult2_2.csv testing/testing_data/mpo_mult2_3.csv			
3	Display	object1->display()	MPO Display test passes	MPO Display test passes	
4	Display	object2->display()	MPO Display test passes	MPO Display test passes	

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/20/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MPO_MultipleFiles_2

Test Description: Testing Functionality of MPO Multiple Files

repo-Team13/Project2/testing/testing_data/mpo_mult1_1.csv
repo-Team13/Project2/testing/testing_data/mpo_mult1_2.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_1.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_2.csv
repo-Team13/Project2/testing/testing_data/mpo_mult2_3.csv

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run test 1	./election testing/test_data/mpo_mult1_1.csv testing/test_data/mpo_mult1_2.csv	MPO Display test passes	MPO Display test passes	
2	Run test 2	./election testing/testing_data/mpo_mult2_1.csv testing/testing_data/mpo_mult2_2.csv testing/testing_data/mpo_mult2_3.csv	MPO Display test passes	MPO Display test passes	

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/20/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MV_MultipleFiles1

Test Description: Multiple MV files can be aggregated into 1 election that is calculated.

repo-Team13/Project2/testing/testing_data/mv_mult1.csv
repo-Team13/Project2/testing/testing_data/mv_mult2.csv
repo-Team13/Project2/testing/testing_data/mv_mult3.csv

Automated: yes _X__ no ___

Results: Pass Fail X

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create MVElectionData Object	testing/testing_data/mv_mult1.csv testing/testing_data/mv_mult2.csv testing/testing_data/mv_mult3.csv			
2	Display	object1->display()	Display test passes		The test is false because there is no MVElectionData file as that is outside the scope of the sprint.
3					
4					

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/20/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MV_MultipleFiles2

Test Description: Testing Functionality of MV Multiple Files

repo-Team13/Project2/testing/testing_data/mv_mult1.csv
repo-Team13/Project2/testing/testing_data/mv_mult2.csv
repo-Team13/Project2/testing/testing_data/mv_mult3.csv

Automated: yes_X__ no ___

Results: Pass ______ Fail __X______

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run test 1	./election testing/testing_data/mv_mult1.csv testing/testing_data/mv_mult2.csv testing/testing_data/mv_mult3.csv	Display test passes		The test is false because there is no MVElectionData file as that is outside the scope of the sprint.
2					

Post condition(s) for Test: The election is calculated as if the data from all files were aggregated into 1 file.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __

Test Date: 04/18/2024

Test Case ID#: EDP_MPO

Name(s) of Testers: Khalid Qasim

Test Description:
Tests and verifies that ElectionDataParser properly creates MPO Elections

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, MPOFilesTest
Functions: ElectionDataParser::create_election

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call ElectionDataParser::create_election	"../testing/test_data/mpo_example_1.csv"	ElectionData pointer not equal to nullptr		
2	Check the election data object	election1	EXPECT_NE(election1, nullptr)	election1	
3	Call ElectionDataParser::create_election for second file	"../testing/test_data/mpo_example_2.csv"	ElectionData pointer not equal to nullptr		
4	Check the election data object	election2	EXPECT_NE(election2, nullptr)	election2	

Post condition(s) for Test:

Project Name: Project 1: Voting System Team# 13

Test Stage: Unit _x_ System __

Test Date: 04/21/2024

Test Case ID#: EDP_MV1

Name(s) of Testers: Michael Mulhall

Test Description:
Tests and verifies that ElectionDataParser properly creates MV Elections when they use one file.

Automated: yes_X__ no ___

Filename: ElectionDataParserUnitTest.cpp

Testname: ElectionDataParserTest, MVFilesTest
Functions: ElectionDataParser::create_election

Results: Pass __X___ Fail________

Preconditions for Test:

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call ElectionDataParser::create_election	"../testing/test_data/mv_example_1.csv"	ElectionData pointer not equal to nullptr		
2	Check the election data object	EXPECT_NE(election, nullptr)	True	True	
3					
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System Team# 13

Test Stage: Unit _x_ System __

Test Date: 04/21/2024

Test Case ID#: EDP_MV2

Name(s) of Testers: Michael Mulhall

Test Description:
Tests and verifies that ElectionDataParser properly creates MV Elections when they use multiple file.

Filename: ElectionDataParserUnitTest.cpp
Testname: ElectionDataParserTest, MVMultipleFilesTest
Functions: ElectionDataParser::create_election

Automated: yes X no

Results: Pass __X___ Fail________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call ElectionDataParser::create_election	"../testing/test_data/mv_mult1_1.csv", "../testing/test_data/mv_mult1_2.csv", "../testing/test_data/mv_mult1_3.csv",	ElectionData pointer not equal to nullptr		
2	Check the election data object	EXPECT_NE(election, nullptr)	True	True	
3					
4					

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __ **Test Date: 04/21/2024**

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

Test Description: Tests if MPOElectionData returns the correct results for an election that uses one ballot csv file.

Filename: MPOElectionDataUnitTest.cpp
Testname: SingleFile
Functions: display()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election1 and Election2 are created using the MPO example ballot files	ElectionData* election1 = ElectionDataParser::create_election("../testing/test_data/mpo_example_1.csv"); ElectionData* election2 = ElectionDataParser::create_el	N/A	N/A	Two separate elections are created.

		ection("../testing/test_data/mpo_example_2.csv");			

2	election1 and 2 are checked to make sure they were correctly created and are not still null pointers	ASSERT_FALSE(election1 == nullptr);
		
		ASSERT_FALSE(election2 == nullptr);
			True	True	

3	Election1 and Election2 are displayed and compared to the expected result	testing::internal::CaptureStdout();
		election1->display();
		output = testing::internal::GetCapturedStdout();
		EXPECT_TRUE(output == "> 3 Pike, D - WINNER\n> 2 Foster, D - WINNER\n2 Borg, R\n1 Jones, R\n1 Smith, I\n0 Deutsch, R\n"
		|| output == "> 3 Pike, D - WINNER\n> 2 Borg, R - WINNER\n2 Foster, D\n1 Jones, R\n1 Smith, I\n0 Deutsch, R\n");
		
		testing::internal::CaptureStdout();
		election2->display();
		output = testing::internal::GetCapturedStdout();
		EXPECT_EQ(output, ">5 Q, D - WINNER\n4 FosterLongLongLongLongLongLongLongName, D\n");
			True	True	MPOElectionData runs as intended.

4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _x_ System __

Test Date: 04/21/2024

Test Case ID#: ED_MPO

Name(s) of Testers: Michael Mulhall

Test Description: Tests if MPOElectionData returns the correct results for an election that uses multiple ballot csv files.

Filename: MPOElectionDataUnitTest.cpp
Testname: MultipleFiles
Functions: display()

Automated: yes X no

Results: Pass __X___ Fail________

Preconditions for Test:

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

1	Election1 and Election2 are created taking in multiple files.	ElectionData* election1 = ElectionDataParser::create_election(std::vector({
		
		"../testing/test_data/mpo_mult1_1.csv",
		
		"../testing/test_data/mpo_mult1_2.csv"
		}));
		ElectionData* election2 = ElectionDataParser::create_election(std::vector({
		
		"../testing/test_data/mpo_mult2_1.csv",
		
		"../testing/test_data/mpo_mult2_2.csv",
		
		"../testing/test_data/mpo_mult2_3.csv"
		}));			

2	Election1 and 2 are checked to see if they are nullptrs or they ran properly.	ASSERT_FALSE(election1 == nullptr); ASSERT_FALSE(election2	True	True	

		== nullptr);			

3	Election1 and Election2 use the display function and the results are compared to the expected results.	testing::internal::CaptureStdout();
		election1->display();
		output = testing::internal::GetCapturedStdout();
		EXPECT_TRUE(output == "> 6 Pike, D - WINNER\n> 3 Foster, D - WINNER\n3 Deutsch, R\n"
		|| output == "> 6 Pike, D - WINNER\n> 3 Deutsch, R - WINNER\n3 Foster, D\n");
		
		testing::internal::CaptureStdout();
		election2->display();
		output = testing::internal::GetCapturedStdout();
		EXPECT_EQ(output, "> 3 Deutsch, R - WINNER\n> 3 Bingus, D - WINNER\n2 Pike, D\n2 Foster, D\n1 D, R\n1 Tingle, L\n");
			True	True	

4					

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/18/2024
Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MPO1
Test Description: Ballot file with 3 parties, 9 votes, 2 seats for Multiple Popularity Only

repo-Team13/Project2/testing/test_data/mpo_example_1.csv

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election is created with correct file name	Election _Data e = Create_Election("mpo_example_1.csv")			
2	Election results are displayed into the terminal	e.display()	Display test passes	Display test passes	
3					
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/18/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MPO2

Test Description: Ballot file with 2 parties and 9 votes for Multiple Popularity Only

repo-Team13/Project2/testing/test_data/mpo_example_2.csv

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election is created with correct file name	Election _Data e = Create_Election("mpo_example_2.csv")			
2	Election results are displayed into the terminal	e.display()	Display test passes	Display test passes	
3					
4					

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/18/2024

Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MPO1

Test Description: Ballot file with 3 parties, 9 votes, 2 seats for Multiple Popularity Only

repo-Team13/Project2/testing/test_data/mpo_example_1.csv

Automated: yes_X__ no ___

Results: Pass __X___ Fail ________

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election is created with correct file name	Election _Data e = Create_Election("mpo_example_1.csv")			

2	Election results are displayed into the terminal	e.display()	Display test passes	Display test passes	
3					
4					

Post condition(s) for Test:

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/21/2024
Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MV1
Test Description: Ballot file with 3 parties and 9 votes for Municipal Voting

repo-Team13/Project2/testing/test_data/mv_example_1.csv

Automated: yes_X__ no ___

Results: Pass Fail X

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election is created with correct file name	Election _Data e = Create_Election("mv_example_1.csv")			
2	Election results are displayed into the terminal	e.display()	Display test passes		The test is false because there is no MVElectionData file as that is outside the scope of the sprint.
3					
4					

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit __ System _X_

Test Date: 04/21/2024
Name(s) of Testers: Michael Mulhall, Grant Oie, Connell Hagan, and Khalid Qasim

Test Case ID#: MV2
Test Description: Ballot file with 3 parties and 9 votes for Municipal Voting

repo-Team13/Project2/testing/test_data/mv_example_2.csv

Automated: yes_X___ no ___

Results: Pass ________ Fail X___

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Election is created with correct file name	Election _Data e = Create_Election("mv_example_2.csv")			
2	Election results are displayed into the terminal	e.display()	Display test passes		The test is false because there is no MVElectionData file as that is outside the scope of the sprint.
3					
4					