Project 1 - Waterfall Developed Voting System

Team 24

Jack Levine (levin520)

Anthony Ross-Sapienza (rosss001)

Philip Siedlecki (sield009)

Xiuyu Yan (yaxx401)

Test Cases

Table of Contents

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

Test Stage: Unit Y System N **Test Date:** 11/15/19

Test Case ID#: LF_01 **Name(s) of Testers: Jack Levine (Levin520)**

Test Description: test LoadFileErrorTests inside testing/file_data_unittest.cc. This test is designed to test the LoadFile(char* filename) against potential error and edge cases.

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

Automated: Y

Results: Pass

Preconditions for Test: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume only errors in test files are those explicitly being tested. Assume CPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a nonexistent file	filename = “test_files/fake_file.csv”	Assert program exits on error: “Failed to open the file: test_files/fake_file.csv”	Program exits on error: “Failed to open the file: test_files/fake_file.csv”	
2	LoadFile(char* filename) for a file with a non .csv extension	filename = “test_files/test_bad_type.txt”	Assert program exits on error: “Failed to open the file: test_files/test_bad_type.txt; File is not of CSV format”	Program exits on error: “Failed to open the file: test_files/test_bad_type.txt; File is not of CSV format”	
3	LoadFile(char* filename) for a file of neither OPL or CPL voting type	filename = “test_files/test_wrong_format.csv”	Assert program exits on error: “Failed to open the file: test_files/test_wrong_format.csv; File should be lead with either OPL or CPL”	Program exits on error: “Failed to open the file: test_files/test_wrong_format.csv; File should be lead with either OPL or CPL”	
4	LoadFile(char* filename) for a large OPL file	filename = “test_files/test_big_OPL.csv”	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Large OPL file test consists of 100 candidates, 250000 ballots, and 11 seats.
5	LoadFile(char* filename) for a large CPL file	filename = “test_files/test_big_CPL.csv”	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Large CPL file test consists of 25 parties, 11 seats, 250000 ballots, and 100 candidates.

Post condition(s) for Test: Program has either exited, as with test steps 1-3, or control has returned to LoadFile(...) caller without exception, as with tests 4-5.

Project Name: Project 1: Voting System   Team#24

Test Stage: Unit Y   System N

Test Date: 11/15/19

Test Case ID#: LF_02

Name(s) of Testers: Jack Levine (Levin520)

Test Description: test LoadFileOPLTest inside testing/file_data_unittest.cc. This test is designed to test that LoadFile(char* filename) correctly fills File_Data members when called on an OPL file.

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

Automated: Y

Results: Pass

Preconditions for Test: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid OPL file	filename = “test_files/test_good_OPL.csv”	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	Test expected_num_seats against num_seats read from file	expected_num_seats = 3; num_seats = *num_seats, as defined in file*	Expect num_seats to be equal to expected_num_seats	num_seats is equal to expected_num_seats.	num_seats initialized using File_Data::get_num_seats_()
3	Test expected_num_ballots against num_ballots read from file	expected_num_ballots = 9; num_ballots = *num_ballots, as defined in file*	Expect num_ballots to be equal to expected_num_ballots	num_ballots is equal to expected_num_ballots.	num_ballots initialized using File_Data::get_num_ballots_()
4	Test expected_num_candidates against num_candidates read from file	expected_num_candidates = 6; num_candidates = *num_candidates, as defined in file*	Expect num_candidates to be equal to expected_num_candidates	num_candidates is equal to expected_num_candidates.	num_candidates initialized using File_Data::get_num_candidates_()

5	Test each expect_candidates against candidates_ read from file. Achieved by comparing each field of expect_candidates against candidates’ fields, as read from file.	ec[0] = Candidate_Data("Pike", 'D', 3, -1);
		ec[1] = Candidate_Data("Foster", 'D', 2, -1);
		ec[2] = Candidate_Data("Deutsch", 'R', 0, -1);
		ec[3] = Candidate_Data("Borg", 'R', 2, -1);
		ec[4] = Candidate_Data("Jones", 'R', 1, -1);
		ec[5] = Candidate_Data("Smith", 'I', 1, -1);
		
		c[i] = *candidates, as defined in file*
			For each candidate_, expect ec[i].name_ will be string equivalent to c[i].name_; expect ec[i].party_ will be equivalent to c[i].party_;
			expect ec[i].votes_ will be equivalent to c[i].votes_;
			expect ec[i].party_rank_ will be equivalent to c[i].party_rank_.
				All fields of expect_candidates are equivalent to their corresponding candidate read from file.	6 dummy candidates created with members initialized to the expected members based on the input file. candidates have been initialized using File_Data::get_candidates_()
					
					ec[i] & c[i] used to abbreviate expect_candidates[i] and

					candidates[i] respectively to improve readability

Post condition(s) for Test: The File_Data caller of LoadFile(filename) has its members correctly set based on the information contained in the file “test_files/test_good_OPL.csv”.

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

Test Stage: Unit Y System N **Test Date: 11/15/19**

Test Case ID#: LoadFile_03 **Name(s) of Testers: Jack Levine (Levin520)**

Test Description: test LoadFileCPLTest inside testing/file_data_unittest.cc. This test is designed to test that LoadFile(char* filename) correctly fills File_Data members when called on a CPL file.

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

Automated: Y

Results: Pass

Preconditions for Test: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume CPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid CPL file	filename = “test_files/test_good_CPL.csv”	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	Test expected_num_parties against num_parties read from file	expected_num_parties = 4; num_parties = *num_parties, as defined in file*	Expect num_parties to be equal to expected_num_parties	num_parties is equal to expected_num_parties.	num_parties initialized using File_Data::get_num_parties_()
3	Test expected_num_seats against num_seats read from file	expected_num_seats = 7; num_seats = *num_seats, as defined in file*	Expect num_seats to be equal to expected_num_seats	num_seats is equal to expected_num_seats.	num_seats initialized using File_Data::get_num_seats_()
4	Test expected_num_ballots against num_ballots read from file	expected_num_ballots = 13; num_ballots = *num_ballots, as defined in file*	Expect num_ballots to be equal to expected_num_ballots	num_ballots is equal to expected_num_ballots.	num_ballots initialized using File_Data::get_num_ballots_()
5	Test expected_num_candidates against num_candidates read from file	expected_num_candidates = 10; num_candidates = *num_candidates, as defined in file*	Expect num_candidates to be equal to expected_num_candidates	num_candidates is equal to expected_num_candidates.	num_candidates initialized using File_Data::get_num_candidates_()
6	Test each expect_parties against parties_ read from file. Achieved by comparing each field of expect_parties	ep[0] = Party_Data('D', 3, 7, 3, ecD); ep[1] = Party_Data('R', 5, 7, 3, ecR); ep[2] = Party_Data('G', 4, 7, 3, ecG); ep[3] = Party_Data('I', 1, 7, 1, ecI);	For each party_, expect ep[i].name_ will be string equivalent to p[i].name_; expect ep[i].num_candidates_	All fields of expect_parties are equivalent to their corresponding parties read from file.	4 dummy parties created with members initialized to the expected members based on the input file. parties have

	against parties' fields, as read from file.	p[i] = *parties, as defined in file*	will be equivalent to p[i].num_candidates_; expect ep[i].votes_ will be equivalent to p[i].votes_;		been initialized using File_Data::get_party_list_() ecX used to abbreviate expect_candidatesX where X represents a party identifier and ep[i] & p[i] used to abbreviate expect_parties[i] and parties[i] respectively to improve readability

6.1	For each party test, test each expect_parties' candidates against parties' candidates read from file. Achieved by comparing each field of expect_parties' candidates against parties' candidates' fields, as read from file.	ecD[0] = Candidate_Data("Pike", 'D', -1, 1);
		ecD[1] = Candidate_Data("Foster", 'D', -1, 2);
		ecD[2] = Candidate_Data("Floyd", 'D', -1, 3);
		ecR[0] = Candidate_Data("Deutsch", 'R', -1, 1);
		ecR[1] = Candidate_Data("Wong", 'R', -1, 2);
		ecR[2] = Candidate_Data("Walters", 'R', -1, 3);
		ecG[0] = Candidate_Data("Jones", 'G', -1, 1);
		ecG[1] = Candidate_Data("Smith", 'G', -1, 2);
		ecG[2] = Candidate_Data("Lewis", 'G', -1, 3);
		ecI[0] = Candidate_Data("Perez", 'I', -1, 1);
		
		party candidates = *party candidates, as defined in file*
			For each party candidate, expect ep[i].c[k].name_ will be string equivalent to ep[i].c[k].name_; expect ep[i].c[k].party_ will be equivalent to ep[i].c[k].party_; expect ep[i].c[k].votes_ will be equivalent to ep[i].c[k].votes_; expect ep[i].c[k].party_rank_ will be equivalent to ep[i].c[k].party_rank_.	All fields of expect_parties' candidates are equivalent to their corresponding parties' candidates read from file.	10 dummy candidates created with members initialized to the expected members based on the input file. These were then passed to ep[i] on construction as ecX as described above.
					
					c[k] used to abbreviate candidates_[k] to improve readability.

Post condition(s) for Test: The File_Data caller of LoadFile(filename) has its members correctly set based on the information contained in the file "test_files/test_good_CPL.csv".

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

Test Stage: Unit_X__ System___ **Test Date: 11/18/2019**

Test Case ID#: CNS_1 **Name(s) of Testers: Xiuyu Yan**

Test Description: A unit test for Calculate_Num_Seats, input is a no-tie opl file.

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

Automated: yes __X__ no___

Results: Pass

Preconditions for Test: est: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume OPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented. Assume Load_File() and Convert_OPL_To_CPL() work properly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid OPL file	filename = "test_files/test_good_OPL.csv"	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	Test quota_	float quota = blank_file_data->get_quota_();	quota to be equal to expect_quota	quota = expect_quota	quota is a private variable in this function, and is set after executing blank_file_data->Calculate_Num_Seats();
3	Test seats for party D	parties[0].seats_	parties[0].seats_to be equal toexpect_seat_D	parties[0].seats = expect_seat_D	Assume Convert_OPL_To_CPL() works correctly.
4	Test seats for party R	parties[1].seats_	parties[1].seats_to be equal toexpect_seat_R	parties[1].seats_ = expect_seat_R	Assume Convert_OPL_To_CPL() works correctly.
5	Test seats for party I	parties[2].seats_	parties[2].seats_to be equal toexpect_seat_I	parties[2].seats = expect_seat_I	Assume Convert_OPL_To_CPL() works correctly.

Post condition(s) for Test:

The seats for each party has been correctly calculated and the data is stored in the party data structure

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

Test Stage: Unit_X__ System___ **Test Date: 11/18/2019**

Test Case ID#: CNS_2 **Name(s) of Testers: Xiuyu Yan**

Test Description: A unit test for Calculate_Num_Seats, input is a cpl file with no ties.

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

Automated: yes _X__ no___

Results: Pass

Preconditions for Test: est: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume OPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented. Assume Load_File() and Convert_OPL_To_CPL() work properly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid OPL file	filename = “test_files/test_good_OPL_2..csv”	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	Test quota_	float quota = blank_file_data->get_quota_();	quota to be equal to expect_quota	quota = expect_quota	quota is a private variable in this function, and is set after executing blank_file_data->Calculate_Num_Seats();
3	Test seats for party D	parties[0].seats_	parties[0].seats_to be equal toexpect_seat_D	parties[0].seats = expect_seat_D	Assume Convert_OPL_To_CPL() works correctly.
4	Test seats for party R	parties[1].seats_	parties[1].seats_to be equal toexpect_seat_R	parties[1].seats_ = expect_seat_R	Assume Convert_OPL_To_CPL() works correctly.
5	Test seats for party I	parties[2].seats_	parties[2].seats_to be equal toexpect_seat_I	parties[2].seats = expect_seat_I	Assume Convert_OPL_To_CPL() works correctly, This party gets a seat because the remainder is 2, which is larger than party D and R(both have remainder of 1)

Post condition(s) for Test:

The seats for each party has been correctly calculated and the data is stored in the party data structure

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

Test Stage: Unit__ System__X_

Test Date: 11/18/2019

Test Case ID#: CNS_3

Name(s) of Testers: Xiuyu Yan

Test Description: A system test for Calculate_Num_Seats, input is an opl file with 3 ties. This test is to check if the program can handle multiple ties.

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

Automated: yes _X__ no___

Results: Pass

Preconditions for Test: est: Assume test files are located in ./test_files/ relative to file_data_unittest.cc. Assume OPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented. Assume Load_File() and Convert_OPL_To_CPL() work properly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid CPL file	filename = "test_files/test_good_CPL_1.csv\0"	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	../build/bin/votingsystem Run the first time	test_files/test_good_OPL_3.csv	D won 2 seats R won 2 seats I won 0 seats	D won 2 seats R won 2 seats I won 0 seats	This is the first possibility of seat allocation. Initial arrangement is (D:1, R:1, J:0, and all three of them has remainder of 2, and 2 seats are still waiting to be allocated), in this case, D and R get seats.
3	../build/bin/votingsystem Run several times	test_files/test_good_OPL_3.csv	D won 1 seats R won 2 seats I won 1 seats	D won 1 seats R won 2 seats I won 1 seats	This is the second possibility of seat allocation. Initial arrangement is (D:1, R:1, J:0, and all three of them has remainder of 2, and 2 seats are still waiting to be

					allocated), in this case, R and I get seats.
4	../build/bin/votingsystem Run several times	test_files/test_good_OPL_3.csv	D won 2 seats R won 1 seats I won 1 seats	D won 2 seats R won 1 seats I won 1 seats	This is the second possibility of seat allocation. Initial arrangement is (D:1, R:1, J:0, and all three of them has remainder of 2, and 2 seats are still waiting to be allocated), in this case, D and I get seats.

Post condition(s) for Test:

The seats for each party has been correctly calculated and the data is stored in the party data structure

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

Test Stage: Unit___ System_X__ **Test Date:** **11/18/2019**

Test Case ID#: CNS_4 **Name(s) of Testers: Xiuyu Yan**

Test Description: A unit test for Calculate_Num_Seats, input is a cpl file with no ties.

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

Automated: yes _X__ no___

Results: Pass

Preconditions for Test: est: Assume test files are located in ./src/test_files. Assume OPL ballot ordering matches order of listed candidates. Assume at least as many seats as ballots. Assume error free ballots. Assume File_Data private member getters are correctly implemented. Assume Load_File() and Convert_OPL_To_CPL() work properly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	LoadFile(char* filename) for a valid OPL file	filename = "test_files/test_good_CPL_3.csv\0"	Assert no exceptions are thrown for LoadFile(filename)	LoadFile(filename) returns without exception.	Loads file data into dummy File_Data object to test members
2	Test quota_	float quota = blank_file_data->get_quota_();	quota to be equal to expect_quota	quota = expect_quota	quota is a private variable in this function, and is set after executing blank_file_data->Calculate_Num_Seats();
3	Test seats for party D	parties[0].seats_	parties[0].seats_to be equal toexpect_seat_D	parties[0].seats = expect_seat_D	Assume Convert_OPL_To_CPL() works correctly. Party D gets a seat since it has remainder of 3, which is the largest remainder.
4	Test seats for party R	parties[1].seats_	parties[1].seats_to be equal toexpect_seat_R	parties[1].seats_ = expect_seat_R	Assume Convert_OPL_To_CPL() works correctly.Party R gets a seat since quota is 4, and it

					gets 8 votes, 8/4 =2.
5	Test seats for party G	parties[2].seats_	parties[2].seats_to be equal toexpect_seat_G	parties[2].seats = expect_seat_I	Assume Convert_OPL_To_CPL() works correctly, so the party data is in decreasing order of votes. This party gets a seat since it gets 4 votes, 4/4 = 0.
6	Test seats for party I	parties[3].seats_	parties[3].seats_to be equal toexpect_seat_I	parties[3].seats = expect_seat_I	Assume Convert_OPL_To_CPL() works correctly, this party doesn’t have a seat, it only has remainder of 2.

Post condition(s) for Test:

The seats for each party has been correctly calculated and the data is stored in the party data structure

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

Test Stage: Unit_X__ System___ **Test Date: 11/18/2019**

Test Case ID#: CNP_1 **Name(s) of Testers: Philip Siedlecki**

Test Description: Check that OPL To CPL calculates the correct number of parties.

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)	3	3	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to calculate the number of parties correctly. (num_parties_ will be set)

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

Test Stage: Unit_X_ System___ **Test Date: 11/18/2019**

Test Case ID#: NR_01 **Name(s) of Testers: Philip Siedlecki**

Test Description: Verify that Convert_OPL_To_CPL correctly sets the number of republicans.

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)	3	3	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to calculate the number of (republican) candidates correctly.

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

Test Stage: Unit_X_ System___ **Test Date: 11/18/2019**

Test Case ID#: ND_01 **Name(s) of Testers: Philip Siedlecki**

Test Description: Verify that Convert_OPL_To_CPL correctly sets the number of Democrats.

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)	2	2	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to calculate the number of (democrats) candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: NI_01
Test Description: Verify that Convert_OPL_To_CPL correctly sets the number of Independents.

Name(s) of Testers: Philip Siedlecki

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)	1	1	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to calculate the number of (Independents) candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: SR_01

Test Description: Verify that Convert_OPL_To_CPL correctly sorts republicans (by votes).

Name(s) of Testers: Philip Siedlecki

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the republican party	f1	"Phil"	"Phil"	"Phil" has the most votes in the google test f1.
3	Check second candidate	f1	"Al"	"Al"	"Al" should be second by vote total.
4	Check third candidate	f1	"Barry"	"Barry"	"Barry" has the least votes in the republican party.

Post condition(s) for Test:

Convert_OPL_To_CPL is known to sort republican candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: SD_01
Test Description: Verify that Convert_OPL_To_CPL correctly sorts democrats (by votes).

Name(s) of Testers: Philip Siedlecki

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the democrat party	f1	“Sally”	“Sally”	
3	Check second candidate	f1	“Tom”	“Tom”	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to sort democrat candidates correctly.

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

Test Stage: Unit_X_ System___ **Test Date: 11/18/2019**

Test Case ID#: SI_01 **Name(s) of Testers: Philip Siedlecki**

Test Description: Verify that Convert_OPL_To_CPL correctly sorts independents (by votes).

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the independent party	f1	"Troy"	"Troy"	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to sort independent candidates correctly.

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

Test Stage: Unit_X_ System___ **Test Date: 11/18/2019**

Test Case ID#: RR_01 **Name(s) of Testers: Philip Siedlecki**

Test Description: Verify that Convert_OPL_To_CPL correctly ranks republicans (by votes).

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the republican party	f1	1	1	
3	Check second candidate	f1	2	2	
4	Check third candidate	f1	3	3	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to rank republican candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: RD_01
Test Description: Verify that Convert_OPL_To_CPL correctly ranks democrats (by votes).

Name(s) of Testers: Philip Siedlecki

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the democrat party	f1	1	1	
3	Check second candidate	f1	2	2	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to rank democrat candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: RI_01

Test Description: Verify that Convert_OPL_To_CPL correctly ranks independents (by votes).

Name(s) of Testers: Philip Siedlecki

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Convert_OPL_To_CPL()	File_Data f1 (google test setup)			
2	Check the first candidate of the independent party	f1	1	1	

Post condition(s) for Test:

Convert_OPL_To_CPL is known to rank independent candidates correctly.

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

Test Stage: Unit_X_ System___

Test Date: 11/18/2019

Test Case ID#: RA_01

Name(s) of Testers: Philip Siedlecki

Test Description: Verify that Convert_OPL_To_CPL fairly breaks ties.

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

Automated: yes_X_ no___

Results: Pass

Preconditions for Test: There must be a valid Candidate_Data array, in OPL format.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Convert_OPL_To_CPL()	File_Data f2 (google test setup)			F2 contains 4 tied candidates (all in the same party)
2	Add each candidate's rank to an array (int ranks)	f2			
3	Repeat steps 1 and 2 (100,000) times.	f2			
4	Find the average rank sum.	f2			
5	Check that each candidate's rank is within 2.5% of the average.	f2	True	True	`EXPECT_LT((ranks[j] - avg) - (avg * 0.05), 0)`

Post condition(s) for Test:

Convert_OPL_To_CPL breaks ties within parties fairly.

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

Test Stage:

Unit_x_ System___ **Test Date:** 11/18/19

Test Case ID#: AF_01 **Name(s) of Testers: Anthony Ross-Sapienza (rosss001)**

Test Description: Test creation of Audit File with a CPL election

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

Automated: yes__ no_X_

Results: Pass

Preconditions for Test: A file containing a valid CPL election data has been created, processed, and the winners have been determined

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Voting_System v->Audit_File()	File_Data data_			
2	Name_File(ofstream out, true)	ofstream object, boolean value	Call the private method Name_File(std::ofsteam& out, bool isAudit)	Method is called correctly	Name_File(ofstream out, true) is a file to create the correct name for the output file
3	Audit_Info(ofstream out)	ofstream object	Call private method Audit_Info(std::ofstream& out), begin printing data	Method is called correctly, data necessary for an audit is added to the file	Audit_Info(ofstream out) begins printing information to the output file
4	Gen_Info(ofstream out, true)	ofstream object, boolean value	Call private method Gen_Info(std::ofstream& out, bool isAudit), print all election information	Method is called correctly, remaining necessary election information is added to the audit file	Gen_Info(ofstream out, true) finishes printing the information relevant to the audit, including the random seed used
5	audit_out.close()	ofstream audit_out	File is closed	File is closed	

Post condition(s) for Test: An accurate Audit File has been created, named appropriately, had the relevant information necessary to replicate the election added to it, and been closed. No part of the File_Data object has been modified.

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

Test Stage:

Unit_x_ System___ **Test Date: 11/18/19**

Test Case ID#: AF_02 **Name(s) of Testers: Anthony Ross-Sapienza (rosss001)**

Test Description: Test creation of Audit File with an OPL election

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

Automated: yes__ no_X_

Results: Pass

Preconditions for Test: A file containing a valid OPL election data has been created, processed, and the winners have been determined

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Voting_System v->Audit_File()	File_Data data_			
2	Name_File(ofstream out, true)	ofstream object, boolean value	Call the private method Name_File(std::ofsteam& out, bool isAudit)	Method is called correctly	Name_File(ofstream out, true) is a file to create the correct name for the output file
3	Audit_Info(ofstream out)	ofstream object	Call private method Audit_Info(std::ofstream& out), begin printing data	Method is called correctly, data necessary for an audit is added to the file	Audit_Info(ofstream out) begins printing information to the output file
4	Gen_Info(ofstream out, true)	ofstream object, boolean value	Call private method Gen_Info(std::ofstream& out, bool isAudit), print all election information	Method is called correctly, remaining necessary election information is added to the audit file. Includes individual candidate votes	Gen_Info(ofstream out, true) finishes printing the information relevant to the audit, including the random seed used
5	audit_out.close()	ofstream audit_out	File is closed	File is closed	

Post condition(s) for Test: An accurate Audit File has been created, named appropriately, had the relevant information necessary to replicate the election added to it, and been closed. No part of the File_Data object has been modified.

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

Test Stage:
Unit_x_ System___

Test Date: 11/18/19

Test Case ID#: MF_01
Test Description: Test creation of Media File with a CPL election

Name(s) of Testers: Anthony Ross-Sapienza (rosss001)

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

Automated: yes__ no_X_

Results: Pass

Preconditions for Test: A file containing a valid CPL election results has been created, processed, and the winners have been determined

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Voting_System v->Media_File()	File_Data data_			
2	Name_File(ofstream out, false)	ofstream object, boolean value	Call the private method Name_File(std::ofsteam& out, bool isAudit)	Method is called correctly	Name_File(ofstream out, false) is a file to create the correct name for the output file
3	Gen_Info(ofstream out, true)	ofstream object, boolean value	Call private method Gen_Info(std::ofstream& out, bool isAudit), print all election information	Method is called correctly, election information is added to the file	Gen_Info(ofstream out, true) adds all of the relevant election information to the media file
4	audit_out.close()	ofstream audit_out	File is closed	File is closed	

Post condition(s) for Test: An accurate Media File has been created, named appropriately, had the results of the election added to it, and been closed. No part of the File_Data object has been modified.

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

Test Stage:
Unit_x_ System___ **Test Date:** 11/18/19

Test Case ID#: MF_02 **Name(s) of Testers: Anthony Ross-Sapienza (rosss001)**

Test Description: Test creation of Media File with an OPL election

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

Automated: yes__ no_X_

Results: Pass

Preconditions for Test: A file containing a valid OPL election results has been created, processed, and the winners have been determined

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Voting_System v->Media_File()	File_Data data_			
2	Name_File(ofstream out, false)	ofstream object, boolean value	Call the private method Name_File(std::ofsteam& out, bool isAudit)	Method is called correctly	Name_File(ofstream out, false) is a file to create the correct name for the output file
3	Gen_Info(ofstream out, true)	ofstream object, boolean value	Call private method Gen_Info(std::ofstream& out, bool isAudit), print all election information	Method is called correctly, election information is added to the file, including the number of votes each candidate received	Gen_Info(ofstream out, true) adds all of the relevant election information to the media file
4	audit_out.close()	ofstream audit_out	File is closed	File is closed	

Post condition(s) for Test: An accurate Media File has been created, named appropriately, had the results of the election added to it, and been closed. No part of the File_Data object has been modified.

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

Test Stage:
Unit_x_ System___ **Test Date: 11/18/19**

Test Case ID#: DS_01 **Name(s) of Testers: Anthony Ross-Sapienza (rosss001)**

Test Description: Test Display_Winners() with any election type

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

Automated: yes__ no_X_

Results: Pass

Preconditions for Test: A file containing election results has been created, processed, and the winners have been determined

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Voting_System v->Display_Winners()	File_Data data_			
2	Party_Data * p_data = data_->get_party_list()	File_Data data_	A reference the party list from the election is made	A reference to the party list is created and stored in p_data	A reference is made because party_list_ is a private variable
3	for loops iterates through p_data	Party_Data p_data	The winners are displayed on the screen	The winners of each party are displayed on the screen. If a party has won no seats, that is also displayed. Parties are listed in the order that they were passed in	Display_Winners() is the same for both OPL and CPL elections

Post condition(s) for Test: The results for the election have been displayed in the console. No part of the File_Data object has been modified.

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

Test Stage: Unit__ System_X_

Test Date: 11/18/19

Test Case ID#: SYS_1

Name(s) of Testers: Anthony Ross-Sapienza

Test Description: System test with wrong file format

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: An incorrectly formatted file is given to test

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program		Prompt to enter filename given	Prompt to enter filename given	
2	Incorrect file format passed in	/test_files/test_wrong_format.csv	Console warning of incorrect file format	Failed to open the file: test_wrong_format.csv; File should be lead with either OPL or CPL	Program closes after warning

Post condition(s) for Test: The program is closed, the file is not processed, and Voting_System is not called

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

Test Stage: Unit__ System_X_ **Test Date: 11/18/19**

Test Case ID#: SYS_2 **Name(s) of Testers: Anthony Ross-Sapienza**

Test Description: Test system with wrong file format

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: An incorrect file format (not .csv) exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program		Program is opened and prompts the user to input a filename	The program is opened and a prompt is displayed on the console	
2	Filename is entered	test_bad_type.txt	Program closes and displays error	Failed to open the file: test_bad_type.txt; File is not of CSV format	
3					
4					
5					

Post condition(s) for Test: Program is closed and the file is not opened

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

Test Stage: Unit__ System_X_

Test Date: 11/18/19

Test Case ID#: SYS_3

Name(s) of Testers: Anthony Ross-Sapienza

Test Description: Test system with file passed in command line

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: A properly formatted election file is opened when opening the program via command line

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program with filename	/test_files/test_good_OPL.csv	File is processed, Audit and Media files are produced, winners are displayed on console	File is processed, Audit and Media files are produced, and winners are displayed on the console all with correct results	

Post condition(s) for Test: Program successfully opens, processes, determines the winners of, outputs correct Audit and Media files for, and displays the winners of an election file

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

Test Stage: Unit__ System_X_ **Test Date: 11/18/19**

Test Case ID#: SYS_4 **Name(s) of Testers: Anthony Ross-Sapienza**

Test Description: Test system with file entered after program is opened

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: A correctly formatted file exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program		Program is opened and prompts the user to input filename	Program is opened and command line prompt for filename is displayed	
2	Enter filename	/test_files/test_good_CPL.csv	File is processed, Audit and Media files are produced, winners are displayed on console	File is processed, Audit and Media files are produced, and winners are displayed on the console all with correct results	

Post condition(s) for Test: Program successfully opens, processes, determines the winners of, outputs correct Audit and Media files for, and displays the winners of an election file

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

Test Stage: Unit__ System_X_

Test Date: 11/18/19

Test Case ID#: SYS_5

Name(s) of Testers: Anthony Ross-Sapienza

Test Description: A large file is tested

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: A correctly formatted file exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program		Program is opened and prompts the user to input filename	Program is opened and command line prompt for filename is displayed	
2	Enter filename	/test_files/test_big_CPL.csv	File is processed, Audit and Media files are produced, winners are displayed on console	File is processed, Audit and Media files are produced, and winners are displayed on the console all with correct results	Process takes less than 5 minutes with more than 100,000 votes

Post condition(s) for Test: Program successfully opens, processes, determines the winners of, outputs correct Audit and Media files for, and displays the winners of an election file.

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

Test Stage: Unit__ System_X_ **Test Date: 11/18/19**

Test Case ID#: SYS_6 **Name(s) of Testers: Anthony Ross-Sapienza**

Test Description: An OPL file is processed

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: A correctly formatted OPL file exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program with filename	/test_files/test_good_OPL_2.csv	File is processed, Audit and Media files are produced, winners are displayed on console	File is processed, Audit and Media files are produced, and winners are displayed on the console all with correct results	The output files correctly show the number of votes that each candidate received, and has ranked each candidate in their respective parties according to votes received

Post condition(s) for Test: Program successfully opens, processes, determines the winners of, outputs correct Audit and Media files for, and displays the winners of an election file.

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

Test Stage: Unit__ System_X_

Test Date: 11/18/19

Test Case ID#: SYS_7

Name(s) of Testers: Anthony Ross-Sapienza

Test Description: A CPL file is processed

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

Automated: yes___ no_X_

Results: Pass

Preconditions for Test: A correctly formatted CPL file exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open program with filename	/test_files/test_good_CPL_1.csv	File is processed, Audit and Media files are produced, winners are displayed on console	File is processed, Audit and Media files are produced, and winners are displayed on the console all with correct results	Output only shows the number of votes each party received and does not change the order of candidates within their respecitve parties

Post condition(s) for Test: Program successfully opens, processes, determines the winners of, outputs correct Audit and Media files for, and displays the winners of an election file.