Project Name: Project 1: Voting System **Team#12**

Test Stage: Unit [] System [X] **Test Date:** 2021-03-14

Test Case ID#: OPL2wayTie

Test Description: Test the functionality of OPLElection system when there is a tie of votes from 2 candidates and 2 parties.

Name(s) of Testers: Luisa Jimenez

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
testing/opl_testfile_party2waytie.csv

Automated: yes [] no [X]

Results: Pass [X] Fail [ ]

Preconditions for Test:
The voting-system should compile and run without errors.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the csv file to test OPL system	testing/opl_testfile_party2waytie.csv	winners: Pike Foster Deutsch	winners: Pike Foster Deutsch	
2					
3					

Post condition(s) for Test:
The OPLElection system correctly chose the winner from an election.

Test Stage: Unit [] System [X]

Test Date: 2021-03-14

Test Case ID#: OPL3wayTie

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPLElection system when there is a tie of votes from 3 candidates and 3 parties.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
testing/opl_testfile_party3waytie.csv

Automated: yes [] no [X]

Results: Pass [X] Fail [ ]

Preconditions for Test:
The voting-system should compile and run without errors.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the csv file to test OPL system	testing/opl_testfile_party3waytie.csv	winners: Pike Foster	winners: Pike Foster	
2					
3					

Post condition(s) for Test:
The OPLElection system correctly chose the winners from an election.

Test Stage: **Unit []** **System [X]**

Test Date: 2021-03-14

Test Case ID#: OPLZeroVote

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPLElection system when there are 0 votes.

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

Automated: **yes [] no [X]**

Results: **Pass [] Fail [X]**

Preconditions for Test:
The voting-system should compile and run without errors.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run the csv file to test OPL system	testing/opl_testfile_zerovote.csv	no winners displayed in the winners section on screen and media report	Floating point Exception.	
2					
3					

Post condition(s) for Test:
The OPLElection system got into a floating point exception error during its execution.

Test Stage: **Unit []** **System [X]**

Test Date: 2021-03-14

Test Case ID#: OPLMoreSeatsThanCand

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPLElection system when there are more seats available than candidates at a party.

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

Automated: **yes []** **no [X]**

Results: **Pass []** **Fail [X]**

Preconditions for Test:
The voting-system should compile and run without errors.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run the csv file to test OPL system	testing/opl_more_seats_than_cand.csv	Seats redistributed to other candidates.	Segmentation Fault	
2					
3					

Post condition(s) for Test:
The OPLElection system Aborted.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: BallotConstructor

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Ballot's constructor.

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

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Construct Ballot instances with preferred candidates and ID numbers assigned to them.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Loop through Ballot instances and check correctness of Ballot IDs.	0, 1, 2, (none)	0, 1, 2, -1	0, 1, 2, -1	Testing usual ID assignment, then testing what happens if no ID is assigned.

Post condition(s) for Test:
Ballots return the correct ID numbers assigned to them.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-13

Test Case ID#: BallotGetChoice

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Ballot's GetChoice and IncrementRank methods.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/ballot_unittest.cc
GetChoice(), IncrementRank()

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
Constructed Ballot instances with preferred candidates and ID numbers assigned to them.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Loop through Ballot instances and check correctness of preferred candidate.	Test data is provided in testing/ballot_unittest.cc.	Expected data is provided in testing/ballot_unittest.cc.	Actual results match the expected data.	
2	Choose the next-preferred candidate on each Ballot instance.	N/A	N/A	N/A	
3	Go to step 1 if there are still preferred candidates listed on any of the Ballot instances.	N/A	N/A	N/A	

Post condition(s) for Test:
Ballot instances return the correct order of preferred candidates assigned to them.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-13

Test Case ID#: CandidateConstructor

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Candidate's constructor.

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

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
Constructed Candidate instances with names and parties assigned to them.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the correctness of the first Candidate instance.	"Rosen", "D"	"Rosen", "D"	"Rosen", "D"	Testing usual assignments.
2	Check the correctness of the second Candidate instance.	"kleinberg", "r"	"kleinberg", "r"	"kleinberg", "r"	Testing an assignment in all lowercase.
3	Check the correctness of the third Candidate instance.	"Chou Chou Chou Chou Chou Chou Chou", "I I I I I I I I I I I I I I I I I"	"Chou Chou Chou Chou Chou Chou Chou", "I I I I I I I I I I I I I I I I I"	"Chou Chou Chou Chou Chou Chou Chou", "I I I I I I I I I I I I I I I I I"	Testing an assignment with long strings and spaces.
4	Check the correctness of the fourth Candidate instance.	"", ""	"", ""	"", ""	Testing empty strings.

Post condition(s) for Test:
Candidate instances return the correct name and party assigned to them.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: CandidateVotes

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Candidate's AddBallotId and RemoveVotes methods.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/candidate_unittest.cc
AddBallotId(), RemoveVotes()

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Constructed Candidate instances with names and parties assigned to them.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Add Ballot IDs and check the correctness of the candidate's total votes.	0, 1, 2, 3, 4, 5	0, 1, 2, 3, 4, 5	0, 1, 2, 3, 4, 5	
2	Remove all of the candidate's votes and check that the candidate's total votes is 0.	0	0	0	
3	Check that the Ballot IDs distributed to the candidate matches the Ballot IDs returned from the candidate.	0, 1	0, 1	0, 1	

Post condition(s) for Test:
The Ballot IDs distributed to the candidate matches the Ballot IDs returned from the candidate.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: PartyConstructor

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Party's constructor.

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

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Constructed Party instances with names assigned to them.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the correctness of Party names.	"D", "r", "I I I I I I I I I I I I I I I I I", ""	"D", "r", "I I I I I I I I I I I I I I I I I", ""	"D", "r", "I I I I I I I I I I I I I I I I I", ""	

Post condition(s) for Test:
Party instances return the correct names assigned to them.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-13

Test Case ID#: PartyAddCandidateIndex

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Party's AddCandidateIndex method.

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

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
Constructed Party instances with names assigned to them.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Add Candidate indices to a Party instance and check the correctness of the number of candidates.	0, 1, 2, 3, 4	0, 1, 2, 3, 4	0, 1, 2, 3, 4	
2	Check the correctness of the Candidate indices in the Party.	2, 3, 5, 7	2, 3, 5, 7	2, 3, 5, 7	

Post condition(s) for Test:
Party instances return the correct number of candidate indices and correct candidate indices.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: PartyAddVote

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of Party's AddVote method.

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

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Constructed Party instances with names assigned to them.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Add votes to a Party instance and check the correctness of the number of votes received.	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7	

Post condition(s) for Test:
Party instances return the correct number of votes received.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-14

Test Case ID#: ElectionLoggerWriteToAuditFile

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of ElectionLogger's WriteToAuditFile method.

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

Automated: **yes [ ]** **no [X]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Constructed ElectionLogger instance which outputs files in the "testing" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Write a usual string to the audit file.	"ElectionLoggerWriteToAuditFile Unit Test"	"ElectionLoggerWriteToAuditFile Unit Test"	"ElectionLoggerWriteToAuditFile Unit Test"	
2	Write an empty string to the audit file.	""	""	""	
3	Write new lines to the audit file.	"\n\n\n\n\n\n\n\n"	8 new lines	8 new lines	
4	Write a string with a to_string done on an integer.	"The following is an integer: " + std::to_string(9929) + "\n"	"The following is an integer: 9929"	"The following is an integer: 9929"	

Post condition(s) for Test:
Audit file and media report are present in the "testing" directory with the specified contents.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-14

Test Case ID#: ElectionLoggerWriteToMediaReport

Name(s) of Testers: Brian Lu

Test Description: Test the functionality of ElectionLogger's WriteToMediaReport method.

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

Automated: **yes [ ]** **no [X]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Constructed ElectionLogger instance which outputs files in the "testing" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Write a usual string to the media report.	"ElectionLoggerWriteToMediaReport Unit Test"	"ElectionLoggerWriteToMediaReport Unit Test"	"ElectionLoggerWriteToMediaReport Unit Test"	
2	Write an empty string to the media report.	""	""	""	
3	Write new lines to the media report.	"\n\n\n\n\n\n\n\n"	8 new lines	8 new lines	
4	Write a string with a to_string done on an integer.	"The following is an integer: " + std::to_string(9931) + "\n"	"The following is an integer: 9931"	"The following is an integer: 9931"	

Post condition(s) for Test:
Audit file and media report are present in the "testing" directory with the specified contents.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: VotingSystemFileNames

Name(s) of Testers: King Yiu Suen

Test Description: Test if the functionality of readFileName().

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

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
A VotingSystem instance is constructed. Two ballot files named ir_testfile.csv and opl_testfile.csv are created and placed in the same directory as the test file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the correctness of the method.	"abc123.csv"	false	false	Testing a file that does not exist.
2	Check the correctness of the method.	"abc123"	false	false	Testing a file that does not have a file extension.
3	Check the correctness of the method.	"abc.vsc"	false	false	Testing a file that has a file extension but is not csv.
4	Check the correctness of the method.	"csv.abc"	false	false	Testing a file whose name contains csv but the file extension is not csv.
5	Check the correctness of the method.	"csv"	false	false	Testing a file whose name contains csv but has no file extension.
6	Check the correctness of the method.	""	false	false	Testing a file whose name is an empty string.
7	Check the correctness of the method.	"testing/ir_testfile.csv"	true	true	Testing a file that exists and is a csv file.
8	Check the correctness of the method.	"testing/opl_testfile.csv"	true	true	Testing a file that exists and is a csv file.

Post condition(s) for Test: A boolean is returned for each test case, indicating whether the input file exists and is in csv format.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-13

Test Case ID#: VotingSystemCsvToData

Name(s) of Testers: Brian Lu

Test Description: Test if the functionality of CsvToData().

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

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
A VotingSystem instance is constructed. Two ballot files named ir_testfile.csv and opl_testfile.csv are created and placed in the same directory as the test file. The content of the files are the same as the two example files provided in the Software Requirements Specification (SRS) instruction.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the correctness of the method.	"testing/ir_testfile.csv"	Test data is provided in testing/votingsystem_unittest.cc.	Actual results match the expected data.	Testing a ballot file in IR format.
2	Check the correctness of the method.	"testing/opl_testfile.csv"	Test data is provided in testing/votingsystem_unittest.cc.	Actual results match the expected data.	Testing a ballot file in OPL format.

Post condition(s) for Test: A 2-dimensional vector named data is created. The (i, j) entry represents the j-th string in i-th line after removing the delimiters “,”, “(”, “)”, “[“ and “]”.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: Constructor

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPLELection's constructor, get_total_candidates, get_total_seats and get_total_ballots methods.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/oplelection_unittest.cc
get_total_candidates(), get_total_seats(), get_total_ballots()

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Information from a csv file is read and the chosen election type is OPL

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check the total number of candidates is correct	6	6	6	
2	Check the total number of seats for the election is	3	3	3	
3	Check the total number of ballots for the election is	9	9	9	

Post condition(s) for Test:
The information from the file for the number of candidates, seats and ballots are correctly stored into the OPL system.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-13

Test Case ID#: DistributeBallotOPL

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPLElection's DistributeBallots methods.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/oplelection_unittest.cc
DistributeBallots(), get_party(), get_total_votes() from Candidates class

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
information from a csv file is read and the chosen election type is OPL

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the total number of ballots each candidate has is correct.	{3,2,0,2,1,1}	{3,2,0,2,1,1}	{3,2,0,2,1,1}	
2	Check the total number of ballots each party has is correct.	{5,3,1}	{5,3,1}	{5,3,1}	

Post condition(s) for Test:
The ballots are distributed to the corresponding candidate and the number of votes per party is set.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: GetQuotaOPL

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPL's GetQuota() method.

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

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Tha ballots for the candidates would have already been distributed and the quota is computed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Computes the quota for the allocation of seats	ballots: 9, seats: 3	3	3	

Post condition(s) for Test:
The quota number is set.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: AllocateSeatsOPL

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPL's AllocateSeats methods.

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

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Information from a csv file is read, the ballots have been distributed and the quota has been computed.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check each party got the right number of seats by using the “largest remainder formula” method	party votes: {5,3,1}, quota: 3	{2,1,0}	{2,1,0}	

Post condition(s) for Test:
The number of seats for each party is set.

Test Stage: **Unit [X]** **System [ ]**

Test Date: 2021-03-13

Test Case ID#: SelectWinnersOPL

Name(s) of Testers: Luisa Jimenez

Test Description: Test the functionality of OPL's SelectWinners method.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/oplelection_unittest.cc
SelectWinners(), is winner() method from Election class.

Automated: **yes [X]** **no [ ]**

Results: **Pass [X]** **Fail [ ]**

Preconditions for Test:
Information from a csv file is read, the ballots have been distributed,the quota has been computed and the seats have been allocated.

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

1	Check the candidate at i-th index is a winner	candidate[0] = true;
		candidate[1] = true;
		candidate[3] = true;
			true
			true
			true
				true
				true
				true	

Post condition(s) for Test:
The candidate at the i-th index is correctly chosen to be a winner.

Test Stage: Unit [X] System [ ]

Test Date: 2021-03-14

Test Case ID#: IRElection Constructor

Name(s) of Testers: Scott Deyo

Test Description: Test the functionality of IRELection's constructor, get_total_candidates and get_total_ballots methods.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
src/irelection_unittest.cc
get_total_candidates(), get_total_ballots()

Automated: yes [X] no [ ]

Results: Pass [X] Fail [ ]

Preconditions for Test:
information from a csv file is read and the chosen election type is IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the total number of candidates is correct	4	4	4	
2	Check the total number of ballots is correct	6	6	6	

Post condition(s) for Test:
The information from the file for the number of candidates and ballots are correctly stored into the IR system.

Test Stage: **Unit [ ]** **System [X]**	**Test Date:** 2021-03-14
Test Case ID#: IRElection Re-/Disribute Ballots **Test Description:** Test the distribution and redistribution of ballots, which can only be done by running the system.	**Name(s) of Testers:** Scott Deyo
Automated: **yes [ ]** **no [X]**	**Indicate where you are storing the tests (what file) and the name of the method/functions being used.** validated by running system; not saved except as audit files DistributeBallots(), RedistributeBallots(), EliminateCandidate()
Results: **Pass [X]** **Fail [ ]**	

Preconditions for Test:
information from a csv file is read and the chosen election type is IR

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Test the 'typical' example ballot	ir_testfile.csv	Rosen wins, other votes are variable (dependent on coin toss)	Rosen wins, other votes are variable (dependent on coin toss)	
2	Test with 3-way tie	ir_testfile_3waytie.csv	Variable winner	Variable winner	
3	Test with zero votes	ir_testfile_zerovote.csv	Error and exit	Error and exit	

Post condition(s) for Test:
There is variability in cases where the votes are tied, but system results that are not dependent on variability are consistent.

Project Name: The project #, name of your system, and the team#

Test Stage: Indicate whether it is a unit test or a system test.

Test Date: The date the test was performed.

Test Case ID#: A unique ID is required. Decide on a naming convention and use numbering. Example: Ballot_Shuffle_1

Name(s) of Testers: List the names of anyone involved in running this test case.

Test Description: Describe briefly the test objective.

Automated: Indicate if the test is completely automated or being checked manually. (If you have methods running the tests and checking results, select "yes". If you are manually checking results, indicate manual by selecting the "no.")

Results: Indicate if the test passed or failed.

Step #: You will be listing the test steps in order. This number is the step number in the process.

Test Step Description: Details of the test step.

Test Data: What the test data will be for this step. Be clear on what the input data will be. If using a specific file, be clear on the name.

Expected Result: What result are you expecting from the program component or system.

Actual Result: What result were returned based on the test.

Post condition for Test: What will be true after the test has been run? Has the state of the system changed in any way?

Notes: Comments and notes for you and your team members.