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

Test Stage: Unit X System __ **Test Date: Nov.16 / 2019**

Test Case ID# : 001 **Name(s) of Testers: Yingjin Zhang, Sunny Qin**

Test Description:
"random_int" method :
Test random integer generator when for integers list, which is called "Flipcoin" in our code to make sure it is a fair call.

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: A tie appears in ballots of parties or candidates when allocating seats.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test Flip_coins for 2 integers Run random generator 1000 times	Test_flipcoin_int.java	Times of 1: around 500 Times of 2: around 500	Times of 1: 499 Times of 2: 501	The result is not the exact number, but in an acceptable range.
2	Test Flip_coins for more than 2 integers Run random generator 1000 times	Test_flipcoin_int.java	Times of 1: around 333 Times of 2: around 333 Times of 3: around 333	Times of 1: 317 Times of 2: 342 Times of 3: 341	The result is not the exact number, but in an acceptable range.
3					

Post condition(s) for Test:

The random generator for integers is a fair call, which means that nearly equal probability for the two elements in the list .

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.

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

Test Stage: Unit _X_ System __ **Test Date: Nov.16/2019**

Test Case ID#: 002 **Name(s) of Testers: Yingjin Zhang , Sunny Qin**

Test Description:
"random_str" method:
Test random integer generator when for a string list, which is called "Flipcoin" in our code to make sure it is a fair call.

Automated: yes___ no X **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
./Test/Test_flipcoin_str.java

Results: Pass X Fail_____

Preconditions for Test: A tie(same ballots) appears when generate results for candidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test flipcoin for two candidates with same votes Run random generator 1000 times	Test_flipcoin_str.java	Times of 'a': around 500 Times of 'b': around 500	Times of 'a': 492 Times of 'b': 508	The result is not the exact same number, but in an acceptable range.
2	Test flipcoin for three more candidates with same votes Run random generator 1000 times	Test_flipcoin_str.java	Times of 'a': around 333 Times of 'b': around 333 Times of 'c': around 333	Times of 'a': 332 Times of 'b': 317 Times of 'c': 351	The result is not the exact same number, but in an acceptable range.

Post condition(s) for Test:
The random generator for string is a fair call, which means that nearly equal probability for each element in the list .

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

Test Stage: Unit X System __ **Test Date: Nov. 16 2019**

Test Case ID#: 003 **Name(s) of Testers: Yingjin Zhang, Sunny Qin**

Test Description:
"allocate_seat" method:
Test allocating seat method to see whether it can allocate correct seats to each party.

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The flipcoin method is well designed, the ballots for parties are parsed correctly, the number of seats has also been correctly read.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test regular seats case without AC or EX	Test_process_allocateseats.java : party 1: 100 votes party 2: 230 votes party 3: 350 votes seats = 7	party 1: 1 party 2: 2 party 3: 4	party 1: 1 party 2: 2 party 3: 4	The right side is the seats number assigned to the corresponding party.
2	Test a tie between two parties occurs, need flip a coin to decide which party	Test_process_allocateseats.java: party 1: 100 votes party 2: 250 votes party 3: 350 votes seats = 7	party 1: 1 party 2: 2 party 3: 4 or party 1: 1 party 2: 3 party 3: 3	party 1: 1 party 2: 2 party 3: 4	The result is random generated. It depends on the random generator result.

3	Test a tie between three parties occurs, need flip a coin to decide which party	Test_process_allocateseats.java: party 1: 270 votes party 2: 270 votes party 3: 270 votes seats = 7	party 1: 2 party 2: 3 party 3: 3 or party 1: 3 party 2: 3 party 3: 2 or party 1: 3 party 2: 2 party 3: 3	party 1: 2 party 2: 3 party 3: 3	The result is random generated. It depends on the random generator result.
4	Test seats > cands case Test more than two rounds allocation case	Test_process_allocateseats.java: party 1: 1 votes, 3 candidates party 2: 2 votes, 3 candidates party 3: 1004 votes, 1 candidate seats = 4	party 1: 1 party 2: 2 party 3: 1	party 1: 1 party 2: 2 party 3: 1	The right side is the seats number assigned to the corresponding party.

Post condition(s) for Test:
The seats are correctly allocated to parties based on the seats allocation algorithm described in instructions.

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

Test Stage: Unit X System __

Test Date: Nov. 16 2019

Test Case ID#: 004

Name(s) of Testers: Yingjin Zhang, Sunny Qin

Test Description:
"display_results" method:
Testing displaying results method to see whether the voting results can be correctly printed to the screen.

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The OPL and CPL work well, and have already generated results.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test display for OPL results	Test_process_display.java	Candidate Party Ballots / Rank b D 60 a D 40 e G 110 d G 90 c G 50 j I 20 i I 90 h I 110 g I 80 f I 50	Candidate Party Ballots / Rank b D 60 a D 40 e G 110 d G 90 c G 50 j I 20 i I 90 h I 110 g I 80 f I 50	In OPL, the last column shows the ballots for each candidate.
2	Test display for CPL results	Test_process_display.java	Candidate Party Ballots / Rank b D 2 a D 1 e G 3 d G 1 c G 2 j I 3 i I 2 h I 1 g I 5 f I 4	Candidate Party Ballots / Rank b D 2 a D 1 e G 3 d G 1 c G 2 j I 3 i I 2 h I 1 g I 5 f I 4	In CPL, the last column shows the rank for each candidate.

Post condition(s) for Test:

Display the results in a table and correctly print it in the terminal.

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

Test Stage: Unit X System __ **Test Date: Nov. 16 2019**

Test Case ID#: 005 **Name(s) of Testers: Yingjin Zhang, Sunny Qin**

Test Description:
"generate_result_CPL" method:
Test generate results method for CPL to see whether it can generate correct results based on candidates' votes and number of seats.

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: Allocates_seats, flipcoin, and parserCPL methods work well.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test regular CPL with regular seats and votes	The regular part in Test_cpl_generateResult.java party_seats[1,2,3] candidats_ballots[]	a : 1 b : null c : 2 d :1 e : null f :null g : null h : 1 i : 2 j : 3	a : 1 b : null c : 2 d :1 e : nul l f :null g : null h : 1 i : 2 j : 3	cadidates_ballots contains each candidate and his/her corresponding rank in the party. The string on the left of the column is the name of the candidate. The number on the right of the column is the rank of the corresponding candidats. a 'null' represents that the candidate is not been elected.
2	Test CPL results with 0 seat assigned for a party	Test_cpl_generateResult.java party_seats[0,2,3] candidats_ballots[]	a : null b : null c : 2 d : 1 e : null	a : null b : nul c : 2 d : 1 e : null	cadidates_ballots contains each candidate and his/her corresponding rank in the party. The string on the left of the column is the name of the

			f : null g : null h : 1 i : 2 j : 3	f : null g : null h : 1 i : 2 j : 3	candidate. The number on the right of the column is the rank of the corresponding candidats. a 'null' represents that the candidate is not been elected.

Post condition(s) for Test:
The voting results are generated and the candidates are correctly picked based on the rank of candidates and number of seats for parties.

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

Test Stage: Unit X System __ **Test Date: Nov. 16 2019**

Test Case ID#: 006 **Name(s) of Testers: Yingjin Zhang, Sunny Qin**

Test Description:
"generate_result_OPL" method:
Test generate results method for OPL to see whether it can generate correct results based on candidates' votes and number of seats.

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: Allocates_seats, flipcoin, and parserOPL methods work well.

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

1	Test regular generate seats with regular data, no AC or EX	The regular part in Test_opl_generateResult.java party_seats[1,2,3] candidats_ballots[]	a : null b : 200 c : 220 d : null e : 330 f : 410 g : 520 h : null i : null j : 330	a : null b : 200 c : 220 d : null e : 330 f : 410 g : 520 h : null i : null j : 330	The string on the left of the column is the name of the candidate. The number on the right of the column is the number of ballots of the corresponding candidats. a 'null' represents that the candidate is not been elected.
2	Test OPL results with 0 seat assigned for a party	Test_opl_generateResult.java: party_seats[0,2,3] candidats_ballots[]	a : null b : null c : 200 d : null e : 330 f : 440 g : 510 h : null i : null j : 330	a : null b : null c : 200 d : null e : 330 f : 440 g : 510 h : null i : null j : 330	The string on the left of the column is the name of the candidate. The number on the right of the column is the number of ballots of the corresponding candidats. a 'null' represents that the candidate is not been elected.
3	Test tie occurred between two candidates with the same votes .	Test_opl_generateResult.java: party_seats[1,2,3] candidats_ballots[]	a : null b : 100 c : 200 d : null e : 330 f : 440 g : 440 h : null i : null j : 330	a : null b : 100 c : 200 d : null e : 330 f : 440 g : 440 h : null i : null j : 330	The string on the left of the column is the name of the candidate. The number on the right of the column is the number of ballots of the corresponding candidats. a 'null' represents that the candidate is not been elected. There are eight expected values since there is a tie in each party, but they are hard to list, so we only list the output one.
4	Test tie occurred between three more candidates with the same votes.	Test_opl_generateResult.java party_seats[1,1,3] candidats_ballots[]	a : null b : 150 c : null d : 200 e : null f : null g : 440 h : null i : 440 j : 530	a : null b : 150 c : null d : 200 e : null f : null g : 440 h : null i : 440 j : 530	The string on the left of the column is the name of the candidate. The number on the right of the column is the number of ballots of the corresponding candidats. a 'null' represents that the candidate is not been elected. There are nine expected values since there is a tie in each party, but they are hard to

					list, so we only list the output one.

Post condition(s) for Test:
The voting results are generated and the candidates are correctly picked based on the ballots of candidates and number of seats for parties.

Project Name: Project 1: Voting System Team# 12

Test Stage: Unit X System __

Test Date: Nov. 16 2019

Test Case ID#: 007

Name(s) of Testers: Xiaohui Chao

Test Description:
Test the "writeSummaryReport" method to see if the election result is saved in the summary file

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The seats are allocated and the winners are generated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test write summary results to file for OPL	The regular part in Test_WriteSummary.java votingType = "OPL"	Election Summary Report The voting type is OPL.	Election Summary Report The voting type is OPL.	Write success

		partyBallots = {Independent Candidate=0, Green=1, Republican=2, Democratic=3, Reform=1} partyNames = {“Democratic”, “Republican”, “Reform”, “Green”, “Independent Candidate”} results = [{Mary=2, Jack=1}, {Sunny=4, Rex=3}, {Bob=6, Tom=5}, {Brandon=8, Lily=7}, {Joe=9, Alice=10}]	The party ballot results are as follows: Party: Independent Candidate, Number of Votes: 0 Party: Green, Number of Votes: 1 Party: Republican, Number of Votes: 2 Party: Democratic, Number of Votes: 3 Party: Reform, Number of Votes: 1 The candidate ballot results are as follows: Party: Democratic, Candidates: Mary: 2 Jack: 1 Party: Republican, Candidates: Sunny: 4 Rex: 3 Party: Reform, Candidates: Bob: 6 Tom: 5 Party: Green, Candidates: Brandon: 8 Lily: 7 Party: Independent Candidate, Candidates: Joe: 9 Alice: 10 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	The party ballot results are as follows: Party: Independent Candidate, Number of Votes: 0 Party: Green, Number of Votes: 1 Party: Republican, Number of Votes: 2 Party: Democratic, Number of Votes: 3 Party: Reform, Number of Votes: 1 The candidate ballot results are as follows: Party: Democratic, Candidates: Mary: 2 Jack: 1 Party: Republican, Candidates: Sunny: 4 Rex: 3 Party: Reform, Candidates: Bob: 6 Tom: 5 Party: Green, Candidates: Brandon: 8 Lily: 7 Party: Independent Candidate, Candidates: Joe: 9 Alice: 10 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	
2	Test write summary results to file for CPL	The regular part in Test_WriteSummary_CPL.java votingType = “CPL” partyBallots = {Independent Candidate=1, Green=2, Republican=6, Democratic=5, Reform=3} partyNames = {“Democratic”, “Republican”, “Reform”, “Green”, “Independent Candidate”}	Election Summary Report The voting type is CPL. The party ballot results are as follows: Party: Independent Candidate, Number of Votes: 1 Party: Green, Number of Votes: 2 Party: Republican, Number of Votes: 6 Party: Democratic, Number of Votes: 5 Party: Reform, Number of Votes: 3 The candidate ballot results are	Election Summary Report The voting type is CPL. The party ballot results are as follows: Party: Independent Candidate, Number of Votes: 1 Party: Green, Number of Votes: 2 Party: Republican, Number of Votes: 6 Party: Democratic, Number of Votes: 5 Party: Reform, Number of Votes: 3 The candidate ballot results are as follows: Party: Democratic, Candidates: Mary: 20 Jack: 10 Party: Republican, Candidates: Sunny: 40 Rex: 30 Party: Reform, Candidates: Bob: 60 Tom:	Write success

		results = [{Mary=20, Jack=10}, {Sunny=40, Rex=30}, {Bob=60, Tom=50}, {Brandon=80, Lily=70}, {Joe=90, Alice=100}]	as follows: Party: Democratic, Candidates: Mary: 20 Jack: 10 Party: Republican, Candidates: Sunny: 40 Rex: 30 Party: Reform, Candidates: Bob: 60 Tom: 50 Party: Green, Candidates: Brandon: 80 Lily: 70 Party: Independent Candidate, Candidates: Joe: 90 Alice: 100 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	50 Party: Green, Candidates: Brandon: 80 Lily: 70 Party: Independent Candidate, Candidates: Joe: 90 Alice: 100 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	

Post condition(s) for Test:

The election summary report is written to the file.

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

Test Stage: Unit X System __

Test Date: Nov. 16 2019

Test Case ID#: 008

Name(s) of Testers: Xiaohui Chao

Test Description:
Test the "writeAuditFile" method to see if the election result is saved in the audit file

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The data is ready to be written into the audit file. If the input data is string, it can be written directly to the audit file. If the input data is integer, it has been transferred to string.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test write string to audit file	The regular part in Test_WriteAudit.java str = CPL path = AuditFile1.txt	CPL	CPL	Write success
2	Test write null string to audit file	The regular part in Test_WriteAudit.java str = null path = AuditFile2.txt			Write success
3	Test write empty string to audit file	The regular part in Test_WriteAudit.java str = “” path = AuditFile3.txt			Write success

Post condition(s) for Test:

The string is written to the audit file.

Project Name: Project 1: Voting System Team# 12

Test Stage: Unit X System __ **Test Date: Nov. 17 2019**

Test Case ID#: 009 **Name(s) of Testers: Xiaohui Chao**

Test Description:
Test the "parse_opl" method to see if the csv file can be correctly parsed

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The CSV file is in the same folder and ready to be parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the parser function for OPL when there is no tie	The data in test_opl_reg.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [Smith,I] 1,,,,,	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=4, Foster=10}, {Deutsch=2, Walters=3, Borg=6}, {Smith=5}]]	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=4, Foster=10}, {Deutsch=2, Walters=3, Borg=6}, {Smith=5}]]	Parse success

		1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 ,,,1,, ,,,,,1 ,,1,,, ,,,,1, ,1,,,, ,,,1,, 1,,,,, ,,,,,1 ,1,,,, ,1,,,, ,,1,,, ,1,,,, ,1,,,, ,,,1,, ,,,,,1 ,1,,,, ,,,1,, ,1,,,,			

		,,,,1, ,,,1,, ,,,,,1 1,,,,, ,1,,,, auditfile = “audit.txt”			
2	Test the parser function for OPL when there is a tie	The data in test_opl_tie.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [Smith,I] 1,,,,, 1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,,	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=5, Foster=9}, {Deutsch=2, Walters=3, Borg=6}, {Smith=5}]]	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=5, Foster=9}, {Deutsch=2, Walters=3, Borg=6}, {Smith=5}]]	Parse success

		,,,,,1 ,,,1,, ,,,,,1 ,,1,,, ,,,,1, ,1,,,, ,,,1,, 1,,,,, ,,,,,1 ,1,,,, ,1,,,, ,,1,,, ,1,,,, 1,,,,, ,,,1,, ,,,,,1 ,1,,,, ,,,1,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 1,,,,, ,1,,,,			

		auditfile = “audit.txt”			
3	Test the parser function for OPL when there is no tie but two candidates have the same name	The data in test_opl_sameName.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [Walters,I] 1,,,,, 1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 ,,,1,, ,,,,,1 ,,1,,, ,,,,1, ,1,,,, ,,,1,, 1,,,,, ,,,,,1 ,1,,,, ,1,,,, ,,1,,, ,1,,,, ,1,,,, ,,,1,, ,,,,,1 ,1,,,, ,,,1,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 1,,,,, ,1,,,, auditfile = “audit.txt”	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=4, Foster=10}, {Deutsch=2, Walters=3, Borg=6}, {Walters=5}]]	[3, 30, [D, R, I], {I=5, R=11, D=14}, [{Pike=4, Foster=10}, {Deutsch=2, Walters=3, Borg=6}, {Walters=5}]]	Parse success

Post condition(s) for Test:
The csv file is correctly parsed

Project Name: Project 1: Voting System

Team# 12

Test Stage: Unit X System __

Test Date: Nov. 18 2019

Test Case ID#: 010

Name(s) of Testers: Xiaohui Chao

Test Description:
Test the "parse_cpl" method to see if the csv file can be correctly parsed

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

Automated: yes___ no X

Results: Pass X Fail_____

Preconditions for Test: The CSV file is in the same folder and ready to be parsed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the parser function for CPL when there is no tie	The data in test_cpl_reg.csv:	3 30 D R	3 30 D R	Parse success

		CPL 3 [D,R,I] 3 30 6 [Pike,D,1] [Foster,D,2] [Deutsch,R,1] [Borg,R,2] [Walters,R,3] [Smith,I,1] 1,, 1,, ,1, ,1, ,,1 1,, ,,1 1,, ,1, ,,1 ,,1 ,1,	I {I=10, R=14, D=6} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}]	I {I=10, R=14, D=6} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}]	

		,1, 1,, ,,1 ,1, ,1, ,,1 ,1, ,1, ,1, ,,1 ,1, ,,1 ,1, ,,1 ,,1 ,1, 1,, ,1, auditfile = “audit.txt”			
2	Test the parser function for CPL when there is a tie	The data in test_cpl_tie.csv: CPL 3 [D,R,I] 3 30 6 [Pike,D,1] [Foster,D,2] [Deutsch,R,1]	3 30 D R I {I=12, R=9, D=9} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}]	3 30 D R I {I=12, R=9, D=9} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}]	Parse success

		[Borg,R,2] [Walters,R,3] [Smith,I,1] 1,, 1,, ,,1 ,1, ,,1 1,, ,,1 1,, ,1, ,,1 ,,1 ,1, ,,1 1,, 1,, ,1, ,1, ,,1 ,1, 1,, ,1, ,,1 ,1, ,,1 ,1, ,,1 1,, ,,1 1,, ,,1 auditfile = “audit.txt”			
3	Test the parser function for CPL when there is no tie but two candidates have the same name	The data in test_cpl_sameName.csv: CPL 3 [D,R,I] 3 30 6	3 30 D R I {I=10, R=14, D=6} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Walters=1}]	3 30 D R I {I=10, R=14, D=6} [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Walters=1}]	Parse success

		[Pike,D,1] [Foster,D,2] [Deutsch,R,1] [Borg,R,2] [Walters,R,3] [Walters,I,1] 1,, 1,, ,1, ,1, ,,1 1,, ,,1 1,, ,1, ,,1 ,,1 ,1, ,1, 1,, ,,1 ,1, ,1, ,,1			

		,1, ,1, ,1, ,,1 ,1, ,,1 ,1, ,,1 ,,1 ,1, 1,, ,1, auditfile = “audit.txt”			

Post condition(s) for Test:

The csv file is correctly parsed

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

Test Stage: System **Test Date: Nov. 16 2019**

Test Case ID#: 011 **Name(s) of Testers: Rex Zhu**

Test Description:

Test a regular CPL file

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

Automated: no

Results: Pass

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java	test_cpl_reg.csv	Party: I, Number of Votes: 10 Party: R, Number of Votes: 14 Party: D, Number of Votes: 6 Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Smith: 1	Election Summary Report The voting type is CPL. The party ballot results are as follows: Party: I, Number of Votes: 10 Party: R, Number of Votes: 14 Party: D, Number of Votes: 6 The candidate ballot results are as follows: Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Smith: 1 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system	test_cpl_reg.csv			
3	Type csv filename	test_cpl_reg.csv			
4	Type filename for saving result	test_cpl_reg.csv			

Post condition(s) for Test:
The result is correct.

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

Test Stage: System **Test Date: Nov. 16 2019**

Test Case ID#: 012 **Name(s) of Testers: Rex Zhu**

Test Description:
Test the CSV file that contains item with the same names

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

Automated: no

Results: Pass

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java	test_cpl_sameName .csv	Party: I, Number of Votes: 10 Party: R, Number of Votes: 14 Party: D, Number of Votes: 6 Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Walters: 1	Election Summary Report The voting type is CPL. The party ballot results are as follows: Party: I, Number of Votes: 10 Party: R, Number of Votes: 14 Party: D, Number of Votes: 6 The candidate ballot results are as follows: Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Walters: 1 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system				
3	Type csv filename				
4	Type filename for saving result				

Post condition(s) for Test:

The result is correct.

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

Test Stage: System

Test Date: Nov. 16 2019

Test Case ID#: 013

Name(s) of Testers: Rex Zhu

Test Description:
Test the CSV that has ties

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

Automated: no

Results: Pass

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java		Party: I, Number of Votes: 12 Party: R, Number of Votes: 9 Party: D, Number of Votes: 9 Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Smith: 1	Election Summary Report The voting type is CPL. The party ballot results are as follows: Party: I, Number of Votes: 12 Party: R, Number of Votes: 9 Party: D, Number of Votes: 9 The candidate ballot results are as follows: Party: D, Candidates: Pike: 1 Party: R, Candidates: Deutsch: 1 Party: I, Candidates: Smith: 1 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system				
3	Type csv filename				
4	Type filename for saving result	test_cpl_tie.csv			

Post condition(s) for Test:

The result is correct.

Project Name: Project 1: Voting System Team# 12

Test Stage: System

Test Date: Nov. 16 2019

Test Case ID#: 014

Name(s) of Testers: Rex Zhu

Test Description:

Test OPL regular CSV file.

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

Automated: no

Results: Pass

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java		Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 Party: D, Candidates: Foster: 10 Party: R, Candidates: Borg: 6 Party: I, Candidates: Smith: 5 number of individual votes.	Election Summary Report The voting type is OPL. The party ballot results are as follows: Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 The candidate ballot results are as follows: Party: D, Candidates: Foster: 10 Party: R, Candidates: Borg: 6 Party: I, Candidates: Smith: 5 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system				
3	Type csv filename				
4	Type filename for saving result	test_opl_reg.csv			

Post condition(s) for Test:

The result is correct.

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

Test Stage: System **Test Date: Nov. 16 2019**

Test Case ID#: 015 **Name(s) of Testers: Rex Zhu**

Test Description:
The CSV file in OPL that has items with the same name

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

Automated: no

Results: Pass

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java	test_opl_sameName	Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 Party: D, Candidates: Foster: 10 Party: R, Candidates: Borg: 6 Party: I, Candidates: Walters: 5	Election Summary Report The voting type is OPL. The party ballot results are as follows: Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 The candidate ballot results are as follows: Party: D, Candidates: Foster: 10 Party: R, Candidates: Borg: 6 Party: I, Candidates: Walters: 5 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system				
3	Type csv filename				
4	Type filename for saving result				

Post condition(s) for Test:

The result is correct.

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

Test Stage: System **Test Date: Nov. 16 2019**

Test Case ID#: 016 **Name(s) of Testers: Rex Zhu**

Test Description:

Test the CSV in OPL with tie

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

Automated: no

Results: Pass _____

Preconditions for Test: The input CSV file is legitimate correct and the program can be compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn on a terminal Change directory to ../src Run javac *.java Run java -d ../testing *.java	test_opl_tie.csv	Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 Party: D, Candidates: Foster: 9 Party: R, Candidates: Borg: 6 Party: I, Candidates: Smith: 5	Election Summary Report The voting type is OPL. The party ballot results are as follows: Party: I, Number of Votes: 5 Party: R, Number of Votes: 11 Party: D, Number of Votes: 14 The candidate ballot results are as follows: Party: D, Candidates: Foster: 9 Party: R, Candidates: Borg: 6 Party: I, Candidates: Smith: 5 For CPL, the number is the candidate's order on the original list. For OPL, the number is the candidate's number of individual votes.	Successful! Audit file is good. Audit files are checked! Here are the audit files: https://github.com/zzdpk2/CSCI5801/tree/master/cplandresult
2	Cd to /testing Run javac *.java Run java Test_system				
3	Type csv filename				
4	Type filename for saving result				

Post condition(s) for Test:

The result is correct.