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

Test Stage: Unit X System __

Test Date: **Nov. 29 2019**

Test Case ID#: 001

Name(s) of Testers: Yingjin Zhang

Test Description:
Test whether Parser can correctly generate parser results for information before ballots part in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./testing/Test_parser_initial.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 read_initial_info methods in Paser.java for parsing OPL voting results file	The data in test_opl_reg.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [Smith,I]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	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,,,,, ,1,,,, auditfile = “audit.txt”			
2	Test the read_initial_info methods in Paser.java for parsing CPL voting results file	The data in test_cpl_reg.csv: 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,,	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	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					

Post condition(s) for Test:

The initial information in csv file is correctly parsed

Test Case ID#: 002

Name(s) of Testers: Yingjin Zhang

Test Description:
Test whether Parser can correctly generate parser results for ballotes in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Test/Test_parser_readBallotsl.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 read_ballots methods in Paser.java for parsing CPL voting results file	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,	{A=4, F=5, E=3, D=6, C=2, B=10}	{A=4, F=5, E=3, D=6, C=2, B=10}	success

		,,,1,, ,,,,,1 1,,,,, ,1,,,,			
2	Test the read_ballots methods in Paser.java for parsing OPL voting results file	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,,,,	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	success
3					

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. 29 2019**

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

Test Description:
Test whether Parser can correctly generate parser results for information before ballots part in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./testing/Test_parser_initial.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 read_initial_info methods in Paser.java for parsing OPL voting results file	The data in test_opl_reg.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [Smith,I] 1,,,,, 1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 ,,,1,,	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	success

		,,,,,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 read_initial_info methods in Paser.java for parsing CPL voting results file	The data in test_cpl_reg.csv: 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, ,1, 1,,	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	success

		,,1 ,1, ,1, ,,1 ,1, ,1, ,1, ,,1 ,1, ,,1 ,1, ,,1 ,,1 ,1, 1,, ,1, auditfile = “audit.txt”			
3					

Post condition(s) for Test:

The initial information in 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#: 004

Name(s) of Testers: Yingjin Zhang

Test Description:

Test whether Parser can correctly generate parser results for ballotes in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Test/Test_parser_readBallotsl.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 read_ballots methods in Paser.java for parsing CPL voting results file	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,,,,	{A=4, F=5, E=3, D=6, C=2, B=10}	{A=4, F=5, E=3, D=6, C=2, B=10}	success
2	Test the read_ballots methods in Paser.java for parsing OPL voting results file	1,,,,, 1,,,,, ,1,,,,	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	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,,,,			
3					

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.16 / 2019**

Test Case ID# : 005 **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#: 006 **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#: 007

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#: 008 **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.jav a 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 f : null g : null h : 1 i : 2 j : 3	a : null b : nul c : 2 d : 1 e : null 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.

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#: 009 **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	a : null b : 200 c : 220 d : null e : 330 f : 410	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.

			g : 520 h : null i : null j : 330	g : 520 h : null i : null j : 330	
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: Dec. 11 2019**

Test Case ID#: 010 **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_X__ no

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 votingType1 = "OPL" path1 = "Summary_OPL.txt"	Party: Democratic, Candidates: Mary Party: Republican, Candidates: Sunny Rex Party: Reform, Candidates: Bob Party: Green, Candidates: Lily Party: Independent Candidate, Candidates: Joe	Party: Democratic, Candidates: Mary Party: Republican, Candidates: Sunny Rex Party: Reform, Candidates: Bob Party: Green, Candidates: Lily Party: Independent Candidate, Candidates: Joe	Write success

		candBallots1 = [{Mary=2, Jack=1}, {Sunny=4, Rex=3}, {Bob=6, Tom=5}, {Brandon=8, Lily=7}, {Joe=9, Alice=10}] partyNames1 = {“Democratic”, “Republican”, “Reform”, “Green”, “Independent Candidate”} numOfSeats1 = 3 results1 = [{Mary=2}, {Sunny=4, Rex=3}, {Bob=6}, {Lily=7}, {Joe=9}]			
2	Test write summary results to file for CPL	The regular part in Test_WriteSummary_CPL.java votingType2 = “CPL” path2 = “Summary_CPL.txt” candBallots2 = [{Mary=2, Jack=1}, {Sunny=4, Rex=3}, {Bob=6, Tom=5}, {Brandon=8, Lily=7}, {Joe=9, Alice=10}] partyNames2 = {“Democratic”, “Republican”, “Reform”, “Green”, “Independent Candidate”} results2 = [{Mary=20, Jack=10}, {Sunny=40, Rex=30}, {Bob=60, Tom=50}, {Brandon=80, Lily=70}, {Joe=90, Alice=100}]	Party: Democratic, Candidates: Mary Jack Party: Republican, Candidates: Sunny Rex Party: Reform, Candidates: Bob Tom Party: Green, Candidates: Brandon Lily Party: Independent Candidate, Candidates: Joe Alice	Party: Democratic, Candidates: Mary Jack Party: Republican, Candidates: Sunny Rex Party: Reform, Candidates: Bob Tom Party: Green, Candidates: Brandon Lily Party: Independent Candidate, Candidates: Joe Alice	Write success

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: **Dec. 10 2019**

Test Case ID#: 011

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_X__ no

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 str1 = CPL path1 = AuditFile1.txt	CPL	CPL	Write success

2	Test write null string to audit file	The regular part in Test_WriteAudit.java str2 = null path2 = AuditFile2.txt			Write success
3	Test write empty string to audit file	The regular part in Test_WriteAudit.java str3 = “” path3 = 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. 29 2019

Test Case ID#: 012

Name(s) of Testers: Yingjin Zhang

Test Description:

Test whether Parser can correctly generate parser results for information before ballots part in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./testing/Test_parser_initial.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 read_initial_info methods in Paser.java for parsing OPL voting results file	The data in test_opl_reg.csv: OPL 3 30 6 [Pike,D] [Foster,D] [Deutsch,R] [Borg,R] [Walters,R] [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	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	success

		1,,,,, ,1,,,, auditfile = “audit.txt”			
2	Test the read_initial_info methods in Paser.java for parsing CPL voting results file	The data in test_cpl_reg.csv: 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, ,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=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	3 30 [D, R, I] [{I=0, R=0, D=0}] [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}] [0, 0, 1, 1, 1, 2] 6 [Pike, Foster, Deutsch, Borg, Walters, Smith]	success
3					

Post condition(s) for Test:

The initial information in 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#: 013

Name(s) of Testers: Yingjin Zhang

Test Description:
Test whether Parser can correctly generate parser results for ballotes in OPLand OPL voting file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
./Test/Test_parser_readBallotsl.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 read_ballots methods	1,,,,,	{A=4, F=5, E=3, D=6, C=2,	{A=4, F=5, E=3, D=6, C=2, B=10}	success

	in Paser.java for parsing CPL voting results file	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,,,,	B=10}		
2	Test the read_ballots methods in Paser.java for parsing OPL voting results file	1,,,,, 1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 ,,,1,, ,,,,,1 ,,1,,, ,,,,1, ,1,,,, ,,,1,, 1,,,,, ,,,,,1 ,1,,,, ,1,,,, ,,1,,, ,1,,,, ,1,,,, ,,,1,, ,,,,,1 ,1,,,,	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	success

		,,,1,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 1,,,,, ,1,,,,			
3					

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. 16 2019

Test Case ID#: 014

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

Test Description:

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

Automated: yes___ no X

Results: Pass X Fail________

Preconditions for Test: The parser for CPL works well and the output data are correct.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Test the read_ballots methods in Paser.java for parsing CPL voting results file	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,,,,	{A=4, F=5, E=3, D=6, C=2, B=10}	{A=4, F=5, E=3, D=6, C=2, B=10}	success
3	Test the read_ballots methods in Paser.java for parsing OPL voting results file	1,,,,, 1,,,,, ,1,,,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 ,,,1,, ,,,,,1 ,,1,,, ,,,,1, ,1,,,, ,,,1,, 1,,,,, ,,,,,1 ,1,,,,	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	[{A=4, B=10}, {F=5, E=3, D=6, C=2}]	success

		,1,,,, ,,1,,, ,1,,,, ,1,,,, ,,,1,, ,,,,,1 ,1,,,, ,,,1,, ,1,,,, ,,,,1, ,,,1,, ,,,,,1 1,,,,, ,1,,,,			
4					

Post condition(s) for Test:

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

Test Stage: Unit X System __

Test Date: Dec. 3, 2019

Test Case ID#: 015

Name(s) of Testers: Xiaohui Chao

Test Description:
Test the "chooser" method to see if the user is able to do a search through the file structure and is able to select a file

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

Automated: yes___ no X

Results: Pass X Fail________

Preconditions for Test: A CSV file is saved in an arbitrary folder on the computer.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test search for a file	A csv file (test_cpl_reg.csv)	The user can search for a file in any folder	The user can search for a file in any folder	Success
2	Test choose a file	A csv file (test_cpl_reg.csv)	The user can choose a file. When the user clicks on the file, the "upload file" button changes from grey to blue. When user clicks on the "upload file" button, the file path is saved.	The user can choose a file. When the user clicks on the file, the "upload file" button changes from grey to blue. The right path is shown on the terminal window.	Success
3	Test the "cancel" button	A csv file (test_cpl_reg.csv)	The user can cancel the operation before he hits the "upload file" button	The file path is not shown on the terminal window	Success
4	Test close the window	No data needed	The window is closed successfully	The window is closed successfully. No error shows up.	Success

Post condition(s) for Test:

The CSV file is searched and the file path is shown on the terminal window.

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

Test Stage: Unit X System __ **Test Date: Dec. 4, 2019**

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

Test Description:

Test the "GUI" class to see if the system is able to generate an interface for user to do a search through the file structure and select a file

Automated: yes___ no X

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

Results: Pass X Fail________

Preconditions for Test: A CSV file is saved in an arbitrary folder on the computer.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test generating a frame	N/A	The system generates a frame with a title "Import voting ballots"	The system generates a frame with a title "Import voting ballots"	Success
2	Test creating a button	N/A	The system creates a button with text "Import from a local csv file"	The system creates a button with text "Import from a local csv file"	Success
3	Test creating the second button	N/A	The system creates the second button with text "Creates new ballots csv file"	The system creates the second button with text "Creates new ballots csv file"	Success
4	Test the response when user clicks on the "Import from a local csv file" button	A csv file (test_cpl_reg.csv)	The system creates a window for user to choose the csv file in any folder. After user choose the file, the system creates a window for user to choose the path and file name for the summary report. After that, the system creates a window for user to choose the path and file name for the audit file. Then, the system calculates the ballots and generate results. At the end, the summary report and audit file are saved.	The system creates a window for user to choose the csv file in any folder. After user choose the file, the system creates a window for user to choose the path and file name for the summary report. After that, the system creates a window for user to choose the path and file name for the audit file. Then, the system calculates the ballots and generate results. At the end, the summary report and audit file are saved.	Success
5	Test the response when user clicks on the "Import from a local csv file" button with a wrong csv file	A csv file in a wrong format (test_cpl_wrongformat.csv), in which the first line is "CP"	The system creates a window for user to choose the csv file in any folder. After user choose the file, the system creates a window for user to choose the path and file name for the summary report. After that, the system creates a window for user to choose the path and file name for the audit file. Then the system stops. Some error messages show up on the terminal window.	The system stops. Some error messages show up on the terminal window. The audit file is saved with "CP" in the first line of the file. The summary report file is not generated.	Pass
6	Test the response when user clicks on the "Import from a	A txt file with the same content (test_cpl_reg_txt.txt)	The system cannot parse the txt file	The system cannot parse the txt file	Success

	local csv file" button with a txt file				

Post condition(s) for Test:
The summary report and audit file are saved in the folders where user wants to save them. The election result is shown on the terminal window.

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

Test Stage: Unit X System __ **Test Date:** **Dec. 10, 2019**

Test Case ID#: 017 **Name(s) of Testers: Xiaohui Chao**

Test Description:
Test the voting system (except the GUI part) to see if the system is able to generate correct summary report file and audit file

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

Automated: yes_X__ no

Results: Pass X Fail________

Preconditions for Test: A CSV file is saved on the computer.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test if the system can handle regular opl voting csv file	A csv file (test_opl_reg.csv) Input voting file name: test_opl_reg.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Smith Audit file: Winners for parties with their total votes:	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Smith Audit file: Winners for parties with their total votes:	Success

			{Foster=10}
			Winners for parties with their total votes: {Borg=6}
			Winners for parties with their total votes:
			{Smith=5}
				{Foster=10}
				Winners for parties with their total votes: {Borg=6}
				Winners for parties with their total votes:
				{Smith=5}	

2	Test if the system can handle regular cpl voting csv file	A csv file (test_cpl_reg.csv) Input voting file name: test_cpl_reg.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Pike=1} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Pike=1} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Success
3	Test if the system can handle opl voting csv file when there is a tie	A csv file (test_opl_tie.csv) Input voting file name: test_opl_tie.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Foster=9} Winners for parties with their total votes: {Borg=6} Winners for parties with their total votes: {Smith=5}	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Foster=9} Winners for parties with their total votes: {Borg=6} Winners for parties with their total votes: {Smith=5}	Success
4	Test if the system can handle cpl voting csv file when there is a tie	A csv file (test_cpl_tie.csv) Input voting file name: test_cpl_tie.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Summary file: Party: D, Candidates: Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Success
5	Test if the system can handle opl voting csv file when two people from different parties have the same name	A csv file (test_opl_sameName.csv) Input voting file name: test_opl_sameName.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Walters Audit file: Winners for parties with their total votes: {Foster=10} Winners for parties with their total votes: {Borg=6} Winners for parties with their total votes: {Walters=5}	Summary file: Party: D, Candidates: Foster Party: R, Candidates: Borg Party: I, Candidates: Walters Audit file: Winners for parties with their total votes: {Foster=10} Winners for parties with their total votes: {Borg=6} Winners for parties with their total votes: {Walters=5}	Success
6	Test if the system can handle cpl voting csv file when two people from different parties have the same name	A csv file (test_cpl_sameName.csv) Input voting file name: test_cpl_sameName.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Walters Audit file: Winners for parties with their total votes: {Pike=1} Winners for parties with their total votes: {Deutsch=1}	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Walters Audit file: Winners for parties with their total votes: {Pike=1} Winners for parties with their total votes: {Deutsch=1}	Success

			Winners for parties with their total votes: {Walters=1}	Winners for parties with their total votes: {Walters=1}	
7	Test if the system can handle cpl voting csv file when two people from the same party have the same name	A csv file (test_cpl_reg_error.csv) Input voting file name: test_cpl_reg_error.csv Output summary file name: summary.txt	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Pike=2} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Summary file: Party: D, Candidates: Pike Party: R, Candidates: Deutsch Party: I, Candidates: Smith Audit file: Winners for parties with their total votes: {Pike=2} Winners for parties with their total votes: {Deutsch=1} Winners for parties with their total votes: {Smith=1}	Success

Post condition(s) for Test:

The summary report and audit file are generated and saved. The election results meet the expectations.

Project Name: Project 1: Voting System

Team# 12

Test Stage: Unit X System __

Test Date: Dec. 11 2019

Test Case ID#: 018

Name(s) of Testers: Rex Zhu

Test Description:

Test initialization function for parsers
./Test/Test_parser_initial

Automated: yes___ no X

Results: Pass X Fail________

Preconditions for Test: The format of input files is correct. The logic of files is correct.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	input test csv	test_cpl_reg.csv"	no exception	no exception	success
3	scan the test csv and audit file	audit_test_initial_cpl.txt	no exception	no exception	success

4	compare the result	buffered strings:
		// cpl
		String exp1 = "[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]";
		// opl
		String exp2 = "[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]";
			"[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]"
			
			"[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]"
				"[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=1, Foster=2}, {Deutsch=1, Walters=3, Borg=2}, {Smith=1}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]"
				
				"[3, 30, [D, R, I], [{I=0, R=0, D=0}], [{Pike=0, Foster=0}, {Deutsch=0, Walters=0, Borg=0}, {Smith=0}], [0, 0, 1, 1, 1, 2], 6, [Pike, Foster, Deutsch, Borg, Walters, Smith]]"
					success

5	check out	true	true	true	

Post condition(s) for Test:
The parsers are successfully initialized.