System Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System] **Test Date:** 2 April 2020

Test Case ID#: **ST_01**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test will focus on the Plurality algorithm and test on the situation when there are 1 seat, 5 candidates and 21 ballots.

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

test_plurality_1_good_data.csv

Automated: Yes __ [✓ No]

Results: [✓ Pass] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app				
2	Enter the ballot files	test_plurality_1_good_data.csv q			
3	Enter the number of seats	1			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Richard Kincaid is the winner and others are losers. The percentage of the

					ballots for each candidate is correctly calculated.
Post condition(s) for Test: An audit .txt file was generated and saved inside the same directory as the voting_app executable					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System __] **Test Date:** 2 April 2020

Test Case ID#: **ST_02**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test will focus on the Plurality algorithm and test on the situation when there are 2 seats, 5 candidates and 21 ballots.

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

test_plurality_1_good_data.csv

Automated: Yes __ [✓ No __]

Results: [✓ Pass __] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app				
2	Enter the ballot files	test_plurality_1_good_data.csv q			
3	Enter the number of seats	2			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Richard Kincaid and Megan Wallace are the winners

					and others are losers. The percentage of the ballots for each candidate is correctly calculated.

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System __] **Test Date:** 2 April 2020

Test Case ID#: **ST_03**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a boundary test for the Voting System. This test will focus on the Plurality algorithm and test on the situation when there are multiple candidates but only 1 ballot and 1 seat.

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

test_plurality_1_ballot_1.csv

Automated: Yes __ [✓ No __]

Results: [✓ Pass __] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run ./voting_app				
2	Enter the ballot files	Test_plurality_1_ballot_1.csv q			
3	Enter the number of seats	1			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Leon Perez is the winner and others are losers. The percentage of

					the ballots for each candidate is correctly calculated.
Post condition(s) for Test: An audit .txt file was generated and saved inside the same directory as the voting_app executable					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System [✓ __] **Test Date:** 2 April 2020

Test Case ID#: **ST_04**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a boundary test for the Voting System. This test will focus on the Plurality algorithm and test on the situation when there is only 1 candidate and 1 seat but multiple ballots.

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

test_plurality_1_candidate_1.csv

Automated: Yes __ No [✓ __]

Results: Pass [✓ __] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run ./voting_app				
2	Enter the ballot files	Test_plurality_1_candidate_1.csv q			
3	Enter the number of seats	1			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true		For the printing result: Robert Tarin is winner and there is no loser. The percentage of the

					ballots for each candidate is correctly calculated.
Post condition(s) for Test: An audit .txt file was generated and saved inside the same directory as the voting_app executable					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System __] **Test Date:** 2 April 2020

Test Case ID#: **ST_05**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test runs the voting system under test mode, takes test_STV_ties.csv as ballot file, 10 as number of seats, STV as voting algorithm, and turns off the shuffle option. There will be ties between candidates J,H,I.

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

test_STV_ties.csv

Automated: Yes __ [✓ No __]

Results: [✓ Pass __] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app -t				
2	Enter name of ballot files	test_STV_ties.csv q			
3	Enter the number of seats	10			
4	Enter type of voting algorithm	STV			
5	Turn off shuffle option	true			
6	Confirm inputs	true			
7	Check if printed results are correct and audit file is generated		true	true	For the printed results: All candidates are winners

8	Repeat steps 1-6				
9	Check if printed results are correct and audit file is generated		true	true	For the printed results: All candidates are winners
10	Check if the order of candidates H,I,J, in losers_ vector for two different elections is different		true	true	

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable, the printed result is correct for both elections, and the order of candidates H,I,J in losers_ vector for two elections is different

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System]

Test Date: 2 April 2020

Test Case ID#: ST_06

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test runs the voting system under test mode, takes test_STV_1_candidate_1.csv as ballot file, 1 as number of seats, STV as voting algorithm, and turns off the shuffle option. The ballot file contains 1 candidate and 100 ballots.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
test_STV_1_candidate_1.csv

Automated: Yes __ [✓ No]

Results: [✓ Pass] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app -t				
2	Enter name of ballot files	test_STV_1_candidate_1.csv q			
3	Enter the number of seats	1			
4	Enter type of voting algorithm	STV			
5	Turn off shuffle option	true			
6	Confirm inputs	true			
7	Check if printed results are correct, audit file is generated ,and the content in audit file is		true	true	For the printed results: The only one candidate is the

	correct				winner, and the losers are empty. For the audit file: In the initial distribution, 51 ballots were assigned to the only candidate, and therefore the candidate won.

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable, the printed result is correct, and the content in audit file is correct

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System __]

Test Date: 2 April 2020

Test Case ID#: ST_07

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test runs the voting system under test mode, takes test_STV_1_ballot_1.csv as ballot file, 10 as number of seats, STV as voting algorithm, and turns off the shuffle option. The ballot file contains 10 candidates and 1 ballot.

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

test_STV_1_ballot_1.csv

Automated: Yes __ [✓ No __]

Results: [✓ Pass __] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app -t				
2	Enter name of ballot files	test_STV_1_ballot_1.csv			
3	Enter the number of seats	10			
4	Enter type of voting algorithm	STV			
5	Turn off shuffle option	true			
6	Confirm inputs	true			
7	Check if printed results are correct, audit file is generated ,and the content in audit file is		true	true	For the printed results: The winners_ vector only

	correct				contains Jimmie Cadorette while the losers_ vector contains all other candidates. All candidates are winners at the end of the election. For the audit file: On top of the printed results being in the audit file, should see that Jimmie has only ballot assigned to him on initial distribution.He is declared winner. Everyone else is declared loser one by one randomly during "redistribution"

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable, the printed result is correct, and the content in audit file is correct

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System ----

Test Date: 2 April 2020

Test Case ID#: **ST_08**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test runs the voting system under test mode, takes test_STV_100000_1.csv, test_STV_100000_2.csv, test_STV_100000_3.csv, test_STV_100000_4.csv, test_STV_100000_5.csv as ballot file, 5 as number of seats, STV as voting algorithm, and turns off the shuffle option. The ballot files contain 10 candidates and 100000 ballots.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
test_STV_100000_1.csv
test_STV_100000_2.csv
test_STV_100000_3.csv
test_STV_100000_4.csv
test_STV_100000_5.csv

Automated: Yes __ No ----

Results: Pass ---- Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app -t				
2	Enter name of ballot files	test_STV_100000_1.csv test_STV_100000_2.csv test_STV_100000_3.csv test_STV_100000_4.csv test_STV_100000_5.csv q			

3	Enter the number of seats	5			
4	Enter type of voting algorithm	STV			
5	Turn off shuffle option	true			
6	Confirm inputs	true			
7	Check if printed results are correct, audit file is generated ,and the content in audit file is correct		true	true	For the printed results: There should be at max 5 Candidates in the winners vector. Then there should be 5 Candidates who win seats starting with the winners in order of victory, followed by losers in order of most recent defeat until all seats are filled (not necessary to fill seats with losers if there are already 5 winners). For the audit file: On top of the printed results stated above, all Candidates in the winners vector should have droop quota Ballots, and all Candidates in the losers vector should have 0 Ballots. At least one new Candidate should be a loser after each distribution round
Post condition(s) for Test:					

An audit .txt file was generated and saved inside the same directory as the voting_app executable, the printed result is correct, and the content in audit file is correct

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System]

Test Date: 2 April 2020

Test Case ID#: **ST_09**

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test will focus on the Plurality algorithm and test on the ties situation when there are 1 seat, 5 candidates and 20 ballots.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
test_plurality_1_ties.csv

Automated: Yes __ [✓ No]

Results: [✓ Pass] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app				
2	Enter the ballot files	test_plurality_1_ties.csv q			
3	Enter the number of seats	1			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Richard Kincaid is the winner and others are losers. The percentage of the

					ballots for each candidate is correctly calculated.
Post condition(s) for Test: An audit .txt file was generated and saved inside the same directory as the voting_app executable					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System ----

Test Date: 2 April 2020

Test Case ID#: ST_10

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test will focus on the Plurality algorithm and test on the ties situation when there are 2 seats, 5 candidates and 20 ballots.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
test_plurality_1_ties.csv

Automated: Yes __ No ----

Results: Pass ---- Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app				
2	Enter the ballot files	test_plurality_1_ties.csv q			
3	Enter the number of seats	2			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			
6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Richard Kincaid and Megan Wallace are the winners and others are

					losers. The percentage of the ballots for each candidate is correctly calculated.

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ [✓ System] **Test Date:** 2 April 2020

Test Case ID#: ST_11

Name(s) of Testers:
Sara Nelson (nels8907)
Brendan Ritchie (ritch167)
Yiwen Xu (xu000515)
Yifan Zhang (zhan4372)

Test Description:
This is a good data test for the Voting System. This test will focus on the Plurality algorithm and test on the situation when there are 1 seat, 10 candidates and 100000 ballots.

Indicate where you are storing the tests (what file) and the name of the method/functions being used.
test_plurality_2_1.csv
test_plurality_2_2.csv
test_plurality_2_3.csv
test_plurality_2_4.csv
test_plurality_2_5.csv

Automated: Yes __ [✓ No]

Results: [✓ Pass] Fail___

Preconditions for Test:
Ballot .csv files are already created and placed in the same directory as the voting_app executable, the .csv files contain no mistakes (perfect play)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run ./voting_app				
2	Enter the ballot files	test_plurality_2_1.csv test_plurality_2_2.csv test_plurality_2_3.csv test_plurality_2_4.csv test_plurality_2_5.csv q			
3	Enter the number of seats	1			
4	Enter the algorithm name	Plurality			
5	Confirm inputs	true			

6	Check if printed results are correct and audit file is generated		true	true	For the printing result: Megan Wallace is the winner and others are losers. The percentage of the ballots for each candidate is correctly calculated.

Post condition(s) for Test:
An audit .txt file was generated and saved inside the same directory as the voting_app executable

Ballot Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __

Test Date: 1 April 2020

Test Case ID#: **UT_ballot_constructor**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the Ballot constructor to make sure that all properties are set to the correct values upon initialization.

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

test_ballot_constructor.cc
Ballot(...)
All getter functions for the member variables that are being checked

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, vector of 1 Candidate objects			
2	Assert getId() returns an int = 1		true	true	
3	Assert getCurrentChoice == 0		true	true	
4	Assert getCandidates() returns a vector of Candidates that have ids equals to the ones passed into the constructor		true	true	

Post condition(s) for Test:
A valid Ballot object has been created and initialized with the proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_ballot_getCandidates**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getCandidates() function in Ballot class to make sure that it returns the candidates_ class member variable correctly.

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

test_ballot_getCandidates.cc
Ballot(...)
getCandidates()
getId()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, vector of 1 Candidate objects			
2	Assert getCandidates() returns an vector of size = 1		true	true	
3	Assert getCandidates() returns a vector of Candidates that have ids equals to the ones passed into the constructor		true	true	

Post condition(s) for Test:
The candidates_ member variable of the Ballot object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_ballot_getCurrentChoice**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getCurrentChoice() function in Ballot class to make sure that it returns the currentChoice_ class member variable correctly.

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

test_ballot_getCurrentChoice.cc
Ballot(...)
getCurrentChoice()

Automated: Yes ---- No __

Results: Pass ---- Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, vector of 1 Candidate objects			
2	Assert getCurrentChoice() returns an int = 0		true	true	

Post condition(s) for Test:
The currentChoice_member variable of the Ballot object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit _] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_ballot_getId**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getId() function in Ballot class to make sure that it returns the id_ class member variable correctly.

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

test_ballot_getId.cc
Ballot(...)
getId()

Automated: [✓ Yes _] No __

Results: [✓ Pass _] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, vector of 1 Candidate objects			
2	Assert getId() returns an int = 1		true	true	

Post condition(s) for Test:
The id_member variable of the Ballot object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_ballot_nextChoice_01**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the nextChoice() function in Ballot class to make sure that it sets the currentChoice_ member variable to next choice.

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

test_ballot_nextChoice.cc
Ballot(...)
nextChoice()
getCurrentChoice()

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, vector of 1 Candidate objects			
2	Call the nextChoice() function using the recently created Ballot object				
3	Assert getCurrentChoice() returns an int = 1		true	true	

Post condition(s) for Test:
The currentChoice_member variable of the Ballot object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_ballot_nextChoice_02**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the nextChoice() function in Ballot class to make sure that it sets the currentChoice_ member variable to next choice.

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

test_ballot_nextChoice.cc
Ballot(...)
nextChoice()
getCurrentChoice()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, empty vector of Candidate objects			
2	Call the nextChoice() function using the recently created Ballot object				
3	Assert getCurrentChoice() returns an int = -1		true	true	

Post condition(s) for Test:
The currentChoice_member variable of the Ballot object is set with proper values

Project Name: Project 1: Voting System

Team# 4

Test Stage: [✓ Unit] System __

Test Date: 1 April 2020

Test Case ID#: UT_ballot_setId

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the setId() function in Ballot class to make sure that it sets the id_ class member variable correctly.

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

test_ballot_constructor.cc
Ballot(...)
setId()
getId()

Automated: [✓ Yes] No __

Results: [✓ Pass] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Ballot object	id = 1, empty vector of Candidate objects			
2	Call setId() to set the id_ member variable of recently created Ballot object to int = 3				
2	Assert getId() returns an int = 3		true	true	

Post condition(s) for Test:
The id_member variable of the Ballot object is set with proper values

Candidate Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_candidate_constructor

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the Candidate constructor to make sure that all properties are set to the correct values upon initialization.

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

test_candidate_constructor.cc
Candidate(...)
All getter functions for the member variables that are being checked

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = “test”			
2	Assert getId() returns an int = 1		true	true	
3	Assert getName() returns a string “Test”		true	true	
4	Assert getBallorForSize() returns an int 0		true	true	
5	Assert getWhenGotFirstBallot() returns an int 0		true	true	
6	Assert getAssignedStatus() returns a bool = false		true	true	

Post condition(s) for Test: A valid Candidate object has been created and initialized with the proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ✓ System __

Test Date: 1 April 2020

Test Case ID#: UT_candidate_getId

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getId() function in Candidate class to make sure that it returns the id_ class member variable correctly.

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

test_candidate_getId.cc
Candidate(...)
getId()

Automated: Yes ✓ No __

Results: Pass ✓ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = “Test”			
2	Assert getId() returns an int = 1		true	true	

Post condition(s) for Test:
The id_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_candidate_getName**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getName() function in Candidate class to make sure that it returns the name_ class member variable correctly.

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

test_candidate_getName.cc
Candidate(...)
getName()

Automated: [✓ Yes] _ No __

Results: [✓ Pass] _ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = "Test"			
2	Assert getName() returns an string = "Test"		true	true	

Post condition(s) for Test:
The name_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_candidate_getAssignedStatus

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getAssignedStatus() function in Candidate class to make sure that it returns the assignedStatus_ class member variable correctly.

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

test_candidate_getAssignedStatus.cc
Candidate(...)
getAssignedStatus()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a Candidate object	id = 1, name = “Test”			
2	Assert getAssignedStatus() returns an bool = false		true	true	

Post condition(s) for Test:
The assignedStatus_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_candidate_getWhenGotFirstBallot

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getWhenGotFirstBallot() function in Candidate class to make sure that it returns the whenGotFirstBallot_ class member variable correctly.

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

test_candidate_getWhenGotFirstBallot.cc
Candidate(...)
setWhenGotFirstBallot()
getWhenGotFirstBallot()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the setWhenGotFirstBallot() function of the recently created Candidate object and set the whenGotFirstBallot_ to int = 1				
3	Assert getWhenGotFirstBallot() returns an int = 1		true	true	

Post condition(s) for Test:
The whenGotFirstBallot_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_candidate_setAssignedStatus

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the setAssignedStatus() function in Candidate class to make sure that it sets the assignedStatus_ class member variable correctly.

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

test_candidate_setAssignedStatus.cc
Candidate(...)
setAssignedStatus()
getAssignedStatus()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the setAssignedStatus() function of the recently created Candidate object and set the assignedStatus_ to bool = true				
3	Assert getAssignedStatus() returns an bool = true		true	true	

Post condition(s) for Test:
The assignedStatus_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ✓ System __ **Test Date:** 1 April 2020

Test Case ID#: UT_candidate_setWhenGotFirstBallot

Name(s) of Testers: Yiwen Xu (xu000515)

Test Description:
Test of the setWhenGotFirstBallot() function in Candidate class to make sure that it sets the whenGotFirstBallot_ class member variable correctly.

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

test_candidate_setWhenGotFirstBallot.cc
Candidate(...)
setWhenGotFirstBallot()
getWhenGotFirstBallot()

Automated: Yes ✓ No __

Results: Pass ✓ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the setWhenGotFirstBallot() function of the recently created Candidate object and set the whenGotFirstBallot_ to int = 1				
3	Assert getWhenGotFirstBallot() returns an int = 1		true	true	

Post condition(s) for Test:
The whenGotFirstBallot_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_candidate_getBallotForSize

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getBallotForSize() function in Candidate class to make sure that it returns the ballotForSize_ class member variable correctly.

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

test_candidate_getBallotForSize.cc
Candidate(...)
getBallotForSize()
addBallot()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the function addBallot() of the recently created object Candidate and add a Ballot into the ballotFor_ list				
3	Assert getBallotForSize() returns an int = 1		true	true	

Post condition(s) for Test:
The ballotForSize_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_candidate_getBallotsFor**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the getBallotsFor() function in Candidate class to make sure that it returns the ballotstFor_ class member variable correctly.

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

test_candidate_getBallotsFor.cc
Candidate(...)
getBallotFor()
addBallot()

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = "Test"			
2	Call the function addBallot() of the recently created object Candidate and add a Ballot into the ballotsFor_ list				
3	Assert getBallotsFor() returns a list of Candidates that have ids in right order		true	true	

Post condition(s) for Test:
The ballotsFor_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_candidate_addBallot**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the addBallot() function in Candidate class to make sure that it sets the ballotstFor_ class member variable correctly.

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

test_candidate_addBallot.cc
Candidate(...)
getBallotFor()
addBallot()
getId()

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the function addBallot() of the recently created object Candidate and add 1000 Ballot into the ballotsFor_ list				
3	Call getBallotsFor() returns a list of Candidates				
4	Assert getBallotsFor() returns a list of Candidates that have ids in right order		true	true	

Post condition(s) for Test:
The ballotsFor_member variable of the Candidate object is set with proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_candidate_removeBallot**

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the removeBallot() function in Candidate class to make sure that it sets the ballotstFor_ class member variable correctly.

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

test_candidate_removeBallot.cc
Candidate(...)
getBallotForSize()
removeBallot()
addBallot()
getId()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a Candidate object	id = 1, name = “Test”			
2	Call the function addBallot() of the recently created object Candidate. And add 2 Ballot objects into the ballotsFor_ list which is a member variable of the Candidate object.				
3	Assert that removeBallot() returns a pointer of the removed Ballot object whose id = 2 and getBallorFotSize = 1		true	true	

Post condition(s) for Test:
The ballotsFor_member variable of the Candidate object is set with proper values

Election Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----_ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_addLoser_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the addLoser(Candidate *loser) function in the election class to make sure that it adds loser to losers_ class member variable correctly.

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

test_election_addLoser.cc
PluralityElection(...)

Automated: Yes ----_ No __

Results: Pass ----_ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects,			
2	Call addLoser(temp)	`Candidate *temp = new Candidate(1, "test")`			
3	Assert losers_ is a vector of size 1, its first candidate has id as 1 and name as “test”		true	true	

Post condition(s) for Test:
The losers_ member variable contains only one candidate with id as 1 and name as “test”

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_addWinner_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the addWinner(Candidate *win) function in the election class to make sure that it adds win to winners_ class member variable correctly.

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

test_election_addWinner.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects,			
2	Call addWinner(temp)	`Candidate *temp = new Candidate(1, "test")`			
3	Assert winners_ is a vector of size 1, its first candidate has id as 1 and name as "test"		true	true	

Post condition(s) for Test:
The winners_ member variable contains only one candidate with id as 1 and name as "test"

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getAuditFilePath_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getAuditFilePath() function in the election class to make sure that it returns an empty string when auditFilePath_ is empty

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

test_election_getAuditFilePath.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects,			
2	Assert auditFilePath_ is an empty string		true	true	

Post condition(s) for Test:
getAuditFilePath() returns an empty string

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getAuditFilePath_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getAuditFilePath() function in the election class to make sure that it returns the correct auditFilePath_ value

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

test_election_getAuditFilePath.cc
PluralityElection(...)
setAuditFilePath()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects,			
2	Setup: Set auditFilePath	auditFilePath_ = “testPath”			
3	Assert auditFilePath_’s value is “testPath”		true	true	

Post condition(s) for Test:
getAuditFilePath() returns “testPath”

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getBallots_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getBallots() function in the election class to make sure that it returns an empty vector when ballots_ is empty

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

test_election_getBallots.cc
PluralityElection(...)

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects,			
2	Assert getBallots() returns an empty vector of Ballots		true	true	

Post condition(s) for Test:
getBallots() returns an empty vector of Ballots

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_election_getBallots_02

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getBallots() function in the election class to make sure that it returns ballots_ vector of Ballots

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

test_election_getBallots.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = vector 1 ballot with id = 1 and empty vector of candidates,			
2	Assert getBallots() returns a vector of ballots with size 1 and it contains a ballot with id = 1and empty vector of candidates		true	true	

Post condition(s) for Test:
getBallots() returns a vector of ballots with size 1 and it contains a ballot with id = 1and empty vector of candidates

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_election_getCandidates_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getCandidates() function in the election class to make sure that it returns an empty vector of candidates

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

test_election_getCandidates.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of ballots,			
2	Assert getCandidates() returns an empty vector of candidates		true	true	

Post condition(s) for Test:
getBallots() returns an empty vector of candidates

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getCandidates_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getCandidates() function in the election class to make sure that it returns the correct value of candidates

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

test_election_getCandidates.cc
PluralityElection(...)

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = vector of 1 candidate with id =1, name = “test” and an empty ballotFor_ vector, bals = empty vector of ballots,			
2	Assert getCandidates() returns a vector of 1 candidate with id = 1, name = “test” and an empty ballotsFor_ vector		true	true	

Post condition(s) for Test:
getCandidates() returns a vector of 1 candidate with id = 1, name = “test” and an empty ballotsFor_ vector

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_election_getLosers_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getLosers() function in the election class to make sure that it returns an empty vector of candidates

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

test_election_getLosers.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Assert getLosers() return empty vector of candidates		true	true	

Post condition(s) for Test:
getLosers() return empty vector of candidates

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getLosers_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getLosers() function in the election class to make sure that it returns the correct value of losers_

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

test_election_getLosers.cc
PluralityElection(...)
addBallot()
addLoser()

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			

2	Setup:
	Call addLoser(tempB)
		Candidate *tempC = new Candidate(1, "loser");
		std::vector<Candidate*> tempCV;
		tempCV.push_back(tempC);
		Ballot *tempB = new Ballot(0, tempCV);			

3	Assert getLosers() return vector of 1 candidate with		true	true	

	candidates id as 1 and name as “loser”, and the candidate contains 1 ballot with id = 0 in ballotsFor_ vector				

Post condition(s) for Test:

getLosers() return vector of 1 candidate with candidates id as 1 and name as “loser”, and the candidate contains 1 ballot with id = 0 in ballotsFor_ vector

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getNumSeats_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getNumSeats() function in the election class to make sure that it returns correct value of numSeats_

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

test_election_getNumSeats.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Assert getNumSeats() returns 1		true	true	

Post condition(s) for Test:
getNumSeats() returns 1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getType_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getType() function in the election class to make sure that it returns an empty string when type_ is empty

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

test_election_getType.cc
PluralityElection(...)

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Assert getType() retuns empty string		true	true	

Post condition(s) for Test:
getType() retuns empty string

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getType_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getType() function in the election class to make sure that it returns the correct value of type_

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

test_election_getType.cc
PluralityElection(...)

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = “test”, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Assert getType() retuns “test”		true	true	

Post condition(s) for Test:
getType() retuns “test”

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getWinners_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getWinners() function in the election class to make sure that it returns an empty vector of candidates when winners_ is empty

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

test_election_getWinners.cc
PluralityElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Assert getWinners() returns an empty vector of candidates		true	true	

Post condition(s) for Test:
getWinners() returns an empty vector of candidates

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_getWinners_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getWinners() function in the election class to make sure that it returns a correct vector of candidates of winners_ class member variable

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

test_election_getWinners.cc
PluralityElection(...)
addBallot();
addWinner();

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			

2	Setup:
	Call addWinners(tempB)
		Candidate *tempC = new Candidate(1, "winner");
		std::vector<Candidate*> tempCV;
		tempCV.push_back(tempC);
		Ballot *tempB = new Ballot(0, tempCV);			

3	Assert getWinners() return vector of 1 candidate with		true	true	

	candidates id as 1 and name as “winner”, and the candidate contains 1 ballot with id = 0 in ballotsFor_ vector				
Post condition(s) for Test: Assert getWinners() return vector of 1 candidate with candidates id as 1 and name as “winner”, and the candidate contains 1 ballot with id = 0 in ballotsFor_ vector					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----__ System __ **Test Date:** 1 April 2020

Test Case ID#: UT_election_setAuditFilePath_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the setAuditFilePath() function in the election class to make sure that it sets auditFilePath_ to an empty string

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

test_election_setAuditFilePath.cc
PluralityElection(...)

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Call setAuditFilePath()	std::string()			
3	Assert auditFilePath_ is an empty string		true	true	

Post condition(s) for Test:
auditFilePath_ is an empty string

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_setAuditFilePath_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the setAuditFilePath() function in the election class to make sure that it sets auditFilePath_ to “testPath”

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

test_election_setAuditFilePath.cc
PluralityElection(...)

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
2	Call setAuditFilePath()	“testPath”			
3	Assert auditFilePath_ is “testPath”		true	true	

Post condition(s) for Test:
auditFilePath_ is “testPath”

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_election_writeToAuditFile_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the writeToAuditFile() function in the election class to make sure that it writes correct text to audit file

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

test_election_writeToAuditFile.cc
PluralityElection(...)
compareFiles(FILE *fp1, FILE *fp2)

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Setup: Create “test1.txt” for comparing	“ Candidates Id, name and their percentage: Winners: Losers: “			
2	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
3	Assert the audit file contains the same content as test1.txt contains		true	true	

Post condition(s) for Test:
the audit file contains the same content as test1.txt contains

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 1 April 2020

Test Case ID#: UT_election_writeToAuditFile_02

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the writeToAuditFile() function in the election class to make sure that it writes correct text to audit file

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

test_election_writeToAuditFile.cc
PluralityElection(...)
compareFiles(FILE *fp1, FILE *fp2)

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Setup: Create “test2.txt” for comparing	“this is text Candidates Id, name and their percentage: Winners: Losers: “			
2	setup: Create a PluralityElection object	type = empty string, seats = 1, cands = empty vector of candidates, bals = empty vector of ballots,			
3	Set auditText_	"this is text"			
4	Assert the audit file contains the same content as test2.txt contains		true	true	

Post condition(s) for Test:
the audit file contains the same content as test2.txt contains

PluralityElection Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_plurality_election_constructor

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection constructor to make sure that all properties are set to the correct values upon initialization.

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

test_plurality_constructor.cc
PluralityElection(...)
All getter functions for the member variables that are being checked

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	type = "Plurality", seats = 3, cands = vector of 4 Candidate objects, bals = vector of 10 Ballot objects,			
2	Assert getType() returns a string “Plurality”				
3	Assert getNumSeats() returns an int = 3		true	true	
4	Assert getCandidates() returns a vector of size = 4		true	true	
5	Assert getCandidates()		true	true	

	returns a vector of Candidates that have ids equals to the ones passed into the constructor				
6	Assert getBallots() returns a vector of size = 10		true	true	
7	Assert getBallots() returns a vector of Ballots that have ids equals to the ones passed into the constructor		true	true	

Post condition(s) for Test:
A valid PluralityElection object has been created and initialized with the proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_plurality_election_calculatePercentage

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection calculatePercentage() function to make sure that all properties are set to the correct values upon initialization.

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

test_plurality_calculatePercentage.cc
PluralityElection(...)
getCandidates()
getId()
calculatePercentage()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a PluralityElection object	seats = 3, cands = vector of 4 Candidate objects, bals = vector of 10 Ballot objects			
2	Distribute the ballots to the candidates				

3	Define a vector of float storing the percentage of ballot for each candidate	float a0 = 2/10;
		float a1 = 3/10;
		float a2 = 5/10;
		float a3 = 0;
		float arr[4] = {a0, a1, a2, a3};
		std::vector<float> x(arr, arr+4);			

4	Call the calculatePercentage() function of the recently created PluralityEleciton				
5	Assert calculatePercentage returns vector of float that is same as the vector we defined in previous step		true	true	
Post condition(s) for Test: The percentage of the ballots for each candidate is calculated properly.					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_plurality_election_getResult_01

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection getResult() function to make sure that all the function returns the correct string which contain the result of the election

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

test_plurality_getResult.cc
PluralityElection(...)
addLoser()
addWinner()
getResult()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	Empty vector winners and empty vector of losers			

2	Create a stringstream test which contains the expected result	this->test << "\nCandidates Id, name and their percentage: ";
		this->test << "\nWinners: ";
		this->test << "\nLosers: ";			

3	Assert getResult() returns a string = test.str()		true	true	

Post condition(s) for Test:

The result is correctly recorded in the getResult() function.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- System __ **Test Date:** 1 April 2020

Test Case ID#: UT_plurality_election_getResult_02

Name(s) of Testers: Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection getResult() function to make sure that all the function returns the correct string which contain the result of the election

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

test_plurality_getResult.cc
PluralityElection(...)
addLoser()
addWinner()
getResult()

Automated: Yes ---- No __

Results: Pass ---- Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	Empty vector winners and empty vector of losers			
2	Create a stringstream test which contains the expected result	this->test << "\nCandidates Id, name and their percentage: "; this->test << "\nWinners: " << "\nId: "; this->test << 0 << "\nName: " << "winner"; this->test << "\nLosers: " << "\nId: "; this->test << 0 << "\nName: "			

		<< "loser";			
3	Call the addWinner() function and add a Candidate object into the vector winners_	Candidate object with id = 0, name = “winner”			
4	Call the addLoser() function and add a Candidate object into the vector losers_	Candidate object with id = 0, name = “loser”			
5	Assert getResult() returns a string = test.str()		true	true	

Post condition(s) for Test:
The result is correctly recorded in the getResult() function.

Team# 4

Test Stage: [✓ Unit __] System __

Test Date: 1 April 2020

Test Case ID#:
UT_plurality_election_sortCandidates_01

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection sortCandidates() function to make sure that the candidates_ member variable is sorted in correct order and also deal with ties situation

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

test_plurality_sortCandidate.cc
PluralityElection(...)
getCandidates()
getId()
getBallots()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a PluralityElection object	seats = 3, cands = vector of 4 Candidate objects, bals = vector of 10 Ballot objects			
2	assign the ballots to the candidates				
3	Call the sortCandidates() function of the recently created PluralityElection				
4	Assert getCandidates returns the vector in which Candidate objects is sorted to correct order based on number of		true	true	

	ballots they have				

Post condition(s) for Test:
The order in the candidates_ member variable is sorted correctly.

Team# 4

Test Stage: Unit __ System __

Test Date: 1 April 2020

Test Case ID#:
UT_plurality_election_sortCandidates_02

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection sortCandidates() function to make sure that the candidates_ member variable is sorted in correct order and also deal with ties situation

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

test_plurality_sortCandidate.cc
PluralityElection(...)
getCandidates()
getId()
getBallots()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	seats = 3, cands = vector of 4 Candidate objects, bals = vector of 10 Ballot objects			
2	assign the ballots to the candidates	Two of the candidates got the same number of ballots(ties)			
3	Call the sortCandidates() function of the recently created PluralityElection				
4	Assert getCandidates returns the vector in		true	true	

	which Candidate objects is sorted to correct order based on number of ballots they have and deal with tie properly				

Post condition(s) for Test:
The order in the candidates_ member variable is sorted correctly.

Team# 4

Test Stage: [✓ Unit __] System __

Test Date: 1 April 2020

Test Case ID#:
UT_plurality_election_runAlgorithm_01

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection runAlgorithm() function to make sure that the plurality algorithm is run correctly including assigning the ballots and adding candidates into winners_ and losers_.

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

test_plurality_runAlgorithm.cc
PluralityElection(...)
runAlgorithm()
getWinners()
getLosers()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	seats = 1, cands = vector of 1 Candidate objects, bals = vector of 1 Ballot objects, all the ballots is assigned to the only one candidate			
2	Call the runAlgorithm() function of the recently created PluralityElection object				
3	Assert the size for getWinners() = 1 and the size for getLosers =		true	true	

	0				
4	Assert getWinners() returns the vector of Candidates that has ids equals to the ones in right order based on the ballots each Candidate got		true	true	

Post condition(s) for Test:
The Plurality Election algorithm is run correctly.

Team# 4

Test Stage: [✓ Unit] System __ **Test Date:** 1 April 2020

Test Case ID#:
UT_plurality_election_runAlgorithm_02

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection runAlgorithm() function to make sure that the plurality algorithm is run correctly including assigning the ballots and adding candidates into winners_ and losers_.

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

test_plurality_runAlgorithm.cc
PluralityElection(...)
runAlgorithm()
getWinners()
getLosers()

Automated: [✓ Yes] No __

Results: [✓ Pass] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	seats = 1, cands = vector of 2 Candidate objects, bals = vector of 5 Ballot objects, 3 ballots are assigned to candidate whose id = 0; 2 ballots are assigned to the candidate whose id = 1			
2	Call the runAlgorithm()				

	function of the recently created PluralityElection object				
3	Assert the size for getWinners() = 1 and the size for getLosers = 1		true	true	
4	Assert getWinners() and getLosers() returns the vector of Candidates that has ids equals to the ones in right order based on the ballots each Candidate got		true	true	

Post condition(s) for Test:
The Plurality Election algorithm is run correctly.

Team# 4

Test Stage: [✓ Unit] __ System __

Test Date: 1 April 2020

Test Case ID#:
UT_plurality_election_runAlgorithm_03

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection runAlgorithm() function to make sure that the plurality algorithm is run correctly including assigning the ballots and adding candidates into winners_ and losers_.

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

test_plurality_runAlgorithm.cc
PluralityElection(...)
runAlgorithm()
getWinners()
getLosers()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object	seats = 2, cands = vector of 2 Candidate objects, bals = vector of 5 Ballot objects, 3 ballots are assigned to candidate whose id = 0; 2 ballots are assigned to the candidate whose id = 1			
2	Call the runAlgorithm() function of the recently				

	created PluralityElection object				
3	Assert the size for getWinners() = 2 and the size for getLosers = 0		true	true	
4	Assert getWinners() and getLosers() returns the vector of Candidates that has ids equals to the ones in right order based on the ballots each Candidate got		true	true	

Post condition(s) for Test:
The Plurality Election algorithm is run correctly.

Team# 4

Test Stage: Unit __ System __

Test Date: 1 April 2020

Test Case ID#:
UT_plurality_election_runAlgorithm_04

Name(s) of Testers:
Yiwen Xu (xu000515)

Test Description:
Test of the PluralityElection runAlgorithm() function to make sure that the plurality algorithm is run correctly including assigning the ballots and adding candidates into winners_ and losers_.

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

test_plurality_runAlgorithm.cc
PluralityElection(...)
runAlgorithm()
getWinners()
getLosers()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a PluralityElection object	seats = 2, cands = vector of 3 Candidate objects, bals = vector of 10 Ballot objects, 3 ballots are assigned to candidate whose id = 0; 3 ballots are assigned to the candidate whose id = 2; 4 ballots are assigned to the candidate			

		whose id = 1			
2	Call the runAlgorithm() function of the recently created PluralityElection object				
3	Assert the size for getWinners() = 2 and the size for getLosers = 1		true	true	
4	Assert getWinners() and getLosers() returns the vector of Candidates that has ids equals to the ones in right order based on the ballots each Candidate got		true	true	

Post condition(s) for Test:
The Plurality Election algorithm is run correctly.

STVElection Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_constructor_01**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the STVElection constructor to make sure that all properties are set to the correct values upon initialization.

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

test_stv_election_constructor.cc
STVElection(...)
All getter functions for the member variables that are being checked

Automated: [✓ Yes] No __

Results: [✓ Pass] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a pointer to an STVElection object	type = "STV", seats = 3, cands = vector of 4 pointers to Candidate objects, bals = vector of 10 pointers to Ballot objects, shuffle = false			
2	Assert getType() returns a string == “STV”		true	**true**	
3	Assert getNumSeats() returns an int == 3		true	**true**	
4	Assert getShuffleStatus() returns a bool == false		true	**true**	
5	Assert getCandidates()		true	**true**	

	returns a vector of size == 4				
6	Assert getCandidates() returns a vector of Candidate* that have ids equals to the ones passed into the constructor		true	**true**	
7	Assert getBallots() returns a vector of size = 10		true	**true**	
8	Assert getBallots() returns a vector of Ballot* that have ids equals to the ones passed into the constructor		true	**true**	
9	Assert getAuditFilePath() returns a string == “”		true	**true**	
10	Assert getDroop() returns an int == 3		true	**true**	
11	Assert getShuffledBallots() returns a vector of size == 10		true	**true**	
12	Assert getShuffledBallots() returns a vector of ints that have values in order 0-9		true	**true**	

Post condition(s) for Test:
A valid STVElection object has been created and initialized with the proper values

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_getDroop_01**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the getDroop() function in the STVElection class to make sure that it returns the droopQuota_ class member variable with the expected value.

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

test_stv_election_getDroop.cc
STVElection(...)
getDroop()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a pointer to an STVElection object	type = "STV", seats = 3, cands = vector of 4 pointers to Candidate objects, bals = vector of 10 pointers to Ballot objects, shuffle = false			
2	Assert getDroop() returns an int == 3		true	**true**	

Post condition(s) for Test:
The droopQuota_ member variable of the STVElection object is set to 3

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __

Test Date: 2 April 2020

Test Case ID#:
UT_stv_election_getShuffledBallots_01

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the getShuffledBallots() function in the STVElection class to make sure that it returns the shuffledBallots_ class member variable with the expected value.

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

test_stv_election_getShuffledBallots.cc
STVElection(...)
getShuffledBallots()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a pointer to an STVElection object	type = "STV", seats = 3, cands = vector of 4 pointers to Candidate objects, bals = vector of 10 pointers to Ballot objects, shuffle = false			
2	Assert getShuffledBallots() returns a vector of size == 10		true	**true**	
3	Assert getShuffledBallots() returns a vector of ints that have values in order 0-9		true	**true**	

Post condition(s) for Test:
The shuffledBallots_ member variable of the STVElection object is set to a vector of 10 ints, 0-9.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_getShuffleStatus_01**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the getShuffleStatus() function in the STVElection class to make sure that it returns the shuffle_ class member variable with the expected value.

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

test_stv_election_getShuffleStatus.cc
STVElection(...)
getShuffleStatus()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create a pointer to an STVElection object	type = "STV", seats = 3, cands = vector of 4 pointers to Candidate objects, bals = vector of 10 pointers to Ballot objects, shuffle = true			
2	Assert getShuffleStatus() returns a bool == true		true	**true**	

Post condition(s) for Test:
The shuffle_ member variable of the STVElection object is set to true.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_01**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 1 seat, 1 Candidate, and 1 Ballot. Checks for exact results. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 1 pointer to a Candidate object with id = 0 and name = “A”	Candidate(0, “A”)			
2	Create 1 pointer to a Ballot object with id = 0, and a vector of Candidate* with one element, which is a pointer to the recently created Candidate*	Ballot(0, [&cands[0]])			
3	Create a pointer to an STVElection object	type = "STV", seats = 1, cands = a vector with 1 pointer to a Candidate object in it (the one just created), bals = a vector with 1 pointer to a Ballot			

		object in it (the one just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns == 1		true	**true**	
8	Assert that the Candidate id of the 0th element of the getWinners() return vector == 0		true	**true**	
9	Assert that the size of the ballotsFor_ vector of Candidate in the getWinners() vector == the droopQuota (1)		true	**true**	
10	Assert that the id of the Ballot object in the ballotsFor_ vector of Candidate in the getWinners() vector == 0		true	**true**	
11	Assert that the vector getLosers() returns is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	
Post condition(s) for Test:					

The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] System __ **Test Date:** 2 April 2020

Test Case ID#: UT_stv_election_runAlgorithm_02

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 1 seat, 2 Candidates, and 5 Ballots. Check for exact results. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: [✓ Yes] No __

Results: [✓ Pass] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 2 pointers to Candidate objects with ids 0 and 1, names “A” and “B”	Candidate(0, “A”), Candidate(1, “B”)			
2	Create 5 pointers to Ballot objects with ids 0-4 and vectors of Candidate* with 1-2 elements, which are pointers to the recently created Candidate*	Ballot(0, [&cands[0], &cands[1]]), Ballot(1, [&cands[1]]), Ballot(2, [&cands[0]]). Ballot(3, [&cands[0], &cands[1]]), Ballot(4, [&cands[0], &cands[1]])			
3	Create a pointer to an STVElection object	type = "STV", seats = 1, cands = a vector with 2 pointers to Candidate objects in it (the ones just			

		created), bals = a vector with 5 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns == 1		true	**true**	
8	Assert that the Candidate id of the 0th element of the getWinners() return vector == 0		true	**true**	
9	Assert that the size of the ballotsFor_ vector of Candidate in the getWinners() vector == the droopQuota (3)		true	**true**	
10	Assert that the id of the Ballot objects in the ballotsFor_ vector of Candidate in the getWinners() vector == 0,2,3 (in order)		true	**true**	
11	Assert that the size of vector getLosers() returns == 1		true	**true**	
12	Assert that the Candidate id of the 0th element of the getLosers() return vector		true	**true**	

	== 1				
13	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_03**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 2 seats, 1 Candidate, (unusual pairing) and 5 Ballots. Check for exact results. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create 1 pointer to a Candidate object with id = 0 and name = “A”	Candidate(0, “A”)			
2	Create 5 pointers to Ballot objects with ids 0-4 and vectors of Candidate* with 1 elements, which are pointers to the recently created Candidate*	Ballot(0, [&cands[0]]), Ballot(1, [&cands[0]]), Ballot(2, [&cands[0]]). Ballot(3, [&cands[0]]), Ballot(4, [&cands[0]])			
3	Create a pointer to an STVElection object	type = "STV", seats = 2, cands = a vector with 1 pointer to a Candidate object in it (the one just created), bals = a vector with 5 pointers to Ballot			

		objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns == 1		true	**true**	
8	Assert that the Candidate id of the 0th element of the getWinners() return vector == 0		true	**true**	
9	Assert that the size of the ballotsFor_ vector of Candidate in the getWinners() vector == the droopQuota (2)		true	**true**	
10	Assert that the id of the Ballot objects in the ballotsFor_ vector of Candidate in the getWinners() vector == 0,1 (in order)		true	**true**	
11	Assert that the vector getLosers() returns is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	
Post condition(s) for Test:					

The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_04**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 10 seats, 10 Candidates, and 11 Ballots. Check for exact results. This test is designed to check both tie scenarios (with Candidates with no Ballots and Candidates with some but equal amounts of Ballots) Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 10 pointers to Candidate objects with ids 0-9, names “A” - “J”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”), Candidate(5, “F”), Candidate(6, “G”), Candidate(7, “H”), Candidate(8, “I”), Candidate(9, “J”)			
2	Create 11 pointers to Ballot objects with ids 0-10 and vectors of Candidate* with 5 elements, which are pointers to the recently created Candidate*s	Ballot(0, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(1, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(2, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]),			

		Ballot(3, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(4, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(5, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(6, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(7, [&cands[0], &cands[1], &cands[2], &cands[3], &cands[4]]), Ballot(8, [&cands[5], &cands[6], &cands[7], &cands[8], &cands[9]]), Ballot(9, [&cands[6], &cands[7], &cands[8], &cands[9], &cands[0]]), Ballot(10, [&cands[4], &cands[5], &cands[6], &cands[7], &cands[8]]),			
3	Create a pointer to an STVElection object	type = "STV", seats = 10, cands = a vector with 10 pointers to Candidate objects in it (the ones just created), bals = a vector with 11 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of		true	**true**	

	vector getWinners() returns == 5				
8	Assert that the Candidate ids of the elements of the getWinners() return vector == 0,1,2,3,5 (in order)		true	**true**	
9	Assert that the Ballot ids of the Ballots in the ballotsFor_ vector of Candidates in the getWinners() vector == 0,1, then 2,3, then 4,5, then 6,7, then 8,10		true	**true**	
10	Assert that the size of vector getLosers() returns == 5		true	**true**	
11	Assert that the vector getLosers() returns contains Candidates whose ballotsFor lists are empty		true	**true**	
12	Assert that the first 3 Candidate ids in the getLosers() return vector are either == 7,8,9 in any order		true	**true**	
13	Assert that the next 2 Candidate ids in the getLosers() return vector are == 4, 6 (in order)		true	**true**	
14	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- __ System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_05**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 1 seat, 5 Candidates, and 100 Ballots. Check for result properties since exact results are difficult to verify. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: Yes ---- __ No __

Results: Pass ---- __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 10 pointers to Candidate objects with ids 0-4, names “A” - “E”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”)			
2	Create 100 pointers to Ballot objects with ids 0-99 and vectors of Candidate* with 3-5 elements, which are pointers to the recently created Candidate*s. These are generated randomly using the helper functions createBallots(cands, total) and inVector(nums, num)	Call createBallots() with the vector of Candidates just created and the int 100 bals = createBallots(cands, 100)			

3	Create a pointer to an STVElection object	type = "STV", seats = 1, cands = a vector with 5 pointers to Candidate objects in it (the ones just created), bals = a vector with 100 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns <= number of seats		true	**true**	
8	Assert that the size of vector getWinners() returns + the size of vector getLosers() returns == number of Candidates		true	**true**	
9	Assert that the size of the ballotsFor_ vector for each Candidate in getWinners() == droopQuota_		true	**true**	
10	Assert that the total number of Ballots for the Candidates in getWinners() <= total number of Ballots cast in the Election		true	**true**	

11	Assert that each Candidate in the getLosers() return vector has a ballotsFor_ list that is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- __ System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_06**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 2 seats, 5 Candidates, and 100 Ballots. Check for result properties since exact results are difficult to verify. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: Yes ---- __ No __

Results: Pass ---- __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 10 pointers to Candidate objects with ids 0-4, names “A” - “E”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”)			
2	Create 100 pointers to Ballot objects with ids 0-99 and vectors of Candidate* with 3-5 elements, which are pointers to the recently created Candidate*s. These are generated randomly using the helper functions createBallots(cands, total) and inVector(nums, num)	Call createBallots() with the vector of Candidates just created and the int 100 bals = createBallots(cands, 100)			

3	Create a pointer to an STVElection object	type = "STV", seats = 2, cands = a vector with 5 pointers to Candidate objects in it (the ones just created), bals = a vector with 100 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns <= number of seats		true	**true**	
8	Assert that the size of vector getWinners() returns + the size of vector getLosers() returns == number of Candidates		true	**true**	
9	Assert that the size of the ballotsFor_ vector for each Candidate in getWinners() == droopQuota_		true	**true**	
10	Assert that the total number of Ballots for the Candidates in getWinners() <= total number of Ballots cast in the Election		true	**true**	

11	Assert that each Candidate in the getLosers() return vector has a ballotsFor_ list that is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- __ System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_07**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 2 seats, 10 Candidates, and 100 Ballots. Check for result properties since exact results are difficult to verify. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: Yes ---- __ No __

Results: Pass ---- __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 10 pointers to Candidate objects with ids 0-9, names “A” - “J”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”), Candidate(5, “F”), Candidate(6, “G”), Candidate(7, “H”), Candidate(8, “I”), Candidate(9, “J”)			
2	Create 100 pointers to Ballot objects with ids 0-99 and vectors of Candidate* with 5-10 elements, which are pointers to the recently created Candidate*s. These are generated randomly using the	Call createBallots() with the vector of Candidates just created and the int 100 bals = createBallots(cands, 100)			

	helper functions createBallots(cands, total) and inVector(nums, num)				
3	Create a pointer to an STVElection object	type = "STV", seats = 2, cands = a vector with 10 pointers to Candidate objects in it (the ones just created), bals = a vector with 100 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns <= number of seats		true	**true**	
8	Assert that the size of vector getWinners() returns + the size of vector getLosers() returns == number of Candidates		true	**true**	
9	Assert that the size of the ballotsFor_ vector for each Candidate in getWinners() == droopQuota_		true	**true**	
10	Assert that the total number of Ballots for the		true	**true**	

	Candidates in getWinners() <= total number of Ballots cast in the Election				
11	Assert that each Candidate in the getLosers() return vector has a ballotsFor_ list that is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_08**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 5 seats, 10 Candidates, and 100 Ballots. Check for result properties since exact results are difficult to verify. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Create 10 pointers to Candidate objects with ids 0-9, names “A” - “J”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”), Candidate(5, “F”), Candidate(6, “G”), Candidate(7, “H”), Candidate(8, “I”), Candidate(9, “J”)			
2	Create 100 pointers to Ballot objects with ids 0-99 and vectors of Candidate* with 5-10 elements, which are pointers to the recently created Candidate*s. These are generated randomly using the	Call createBallots() with the vector of Candidates just created and the int 100 bals = createBallots(cands, 100)			

	helper functions createBallots(cands, total) and inVector(nums, num)				
3	Create a pointer to an STVElection object	type = "STV", seats = 5, cands = a vector with 10 pointers to Candidate objects in it (the ones just created), bals = a vector with 100 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns <= number of seats		true	**true**	
8	Assert that the size of vector getWinners() returns + the size of vector getLosers() returns == number of Candidates		true	**true**	
9	Assert that the size of the ballotsFor_ vector for each Candidate in getWinners() == droopQuota_		true	**true**	
10	Assert that the total number of Ballots for the		true	**true**	

	Candidates in getWinners() <= total number of Ballots cast in the Election				
11	Assert that each Candidate in the getLosers() return vector has a ballotsFor_ list that is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 2 April 2020

Test Case ID#: **UT_stv_election_runAlgorithm_09**

Name(s) of Testers:
Brendan Ritchie (ritch167)

Test Description:
Test of the runAlgorithm() function in the STVElection class on an STV election with 3 seats, 10 Candidates, and 100,000 Ballots. Check for result properties since exact results are difficult to verify. This is a load test. Also an indirect test of the redistribute() method as this is a private method that is used in the running of the STV algorithm.

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

test_stv_election_runAlgorithm.cc
STVElection(...)
Ballot(...)
Candidate(...)
runAlgorithm()
redistribute()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create 10 pointers to Candidate objects with ids 0-9, names “A” - “J”	Candidate(0, “A”), Candidate(1, “B”), Candidate(2, “C”), Candidate(3, “D”), Candidate(4, “E”), Candidate(5, “F”), Candidate(6, “G”), Candidate(7, “H”), Candidate(8, “I”), Candidate(9, “J”)			
2	Create 100,000 pointers to Ballot objects with ids 0-99,999 and vectors of Candidate* with 5-10 elements, which are pointers to the recently created Candidate*s. These are generated randomly using the	Call createBallots() with the vector of Candidates just created and the int 100000 bals = createBallots(cands, 100000)			

	helper functions createBallots(cands, total) and inVector(nums, num)				
3	Create a pointer to an STVElection object	type = "STV", seats = 3, cands = a vector with 10 pointers to Candidate objects in it (the ones just created), bals = a vector with 100,000 pointers to Ballot objects in it (the ones just created), shuffle = false			
4	Create a time_t variable to store the current time (time(NULL))	start = time(null)			
5	Call the runAlgorithm() function using the recently created STVElection object				
6	Create a double variable to store the time difference between before and after runAlgorithm() ran	minutes = difftime(time(NULL), start) / 60			
7	Assert that the size of vector getWinners() returns <= number of seats		true	**true**	
8	Assert that the size of vector getWinners() returns + the size of vector getLosers() returns == number of Candidates		true	**true**	
9	Assert that the size of the ballotsFor_ vector for each Candidate in getWinners() == droopQuota_		true	**true**	
10	Assert that the total number of Ballots for the		true	**true**	

	Candidates in getWinners() <= total number of Ballots cast in the Election				
11	Assert that each Candidate in the getLosers() return vector has a ballotsFor_ list that is empty		true	**true**	
12	Assert that the time it took for runAlgorithm() to run (minutes) was less than or equal to 5 minutes		true	**true**	

Post condition(s) for Test:
The winners_ and losers_ member variables of the STVElection class contain the winning and losing Candidate objects according to the STV algorithm (winners of seats are determined in getResults()) and an audit file was generated with the audit trail of the election.

Project Name: Project 1: Voting System

Team# 4

Test Stage: Unit [✓ _] System __

Test Date: 1 April 2020

Test Case ID#: UT_stv_election_calculateDroop_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the calculateDroop() function in the STVElection class to make sure that it calculates droop quota correctly

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

test_stv_election_calculateDroop.cc
STVElection(...)
getDroop()

Automated: Yes [✓ _] No __

Results: Pass [✓ _] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballot objects, Shuffle = false			
2	Call calculateDroop()				
3	Assert getDroop() returns 1		true	true	

Post condition(s) for Test:
droopQuota_ class member variable == 1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_stv_election_calculateDroop_02

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the calculateDroop() function in the STVElection class to make sure that it calculates droop quota correctly

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

test_stv_election_calculateDroop.cc
STVElection(...)
getDroop()

Automated: [✓ Yes _] No __

Results: [✓ Pass _] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = vector of 1 Ballots object, Shuffle = false			
2	Call calculateDroop()				
3	Assert getDroop() returns 1		true	true	

Post condition(s) for Test:
droopQuota_ class member variable == 1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 1 April 2020

Test Case ID#: UT_stv_election_getResults_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getResult() function in the STVElection class to make sure that it returns a correct string

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

test_stv_election_getResults.cc
STVElection(...)
addWinner()
addLoser()

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballots objects, Shuffle = false			

2	Setup:
	Set test string stream
		this->test << "\nWinners vector:";
		this->test << "\nLosers vector:";
		this->test << "\nWinners in Order: ";
		this->test << "\nLosers in Order: ";			

3	Assert getReselts() retuns the same string as test contains		true	true	

Post condition(s) for Test:

getReselts() retuns the same string as test contains

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----__ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_getResults_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getResult() function in the STVElection class to make sure that it returns a correct string

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

test_stv_election_getResults.cc
STVElection(...)
addWinner()
addLoser()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a STVElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballots objects, Shuffle = false			
2	Setup: Call addWinner()	new Candidate(0, "winner")			
3	Setup: Call addLoser()	new Candidate(0, "loser")			

4	Setup:
	Set test string stream
		this->test << "\nWinners vector:";
		this->test << "\n0.\tId: 0\tName: winner";
		this->test << "\nLosers vector:";
		this->test << "\n0.\tId: 0\tName: loser";			

		this->test << "\nWinners in Order: ";
		this->test << "\n0.\tId: 0\tName: winner";
		this->test << "\nLosers in Order: ";
		this->test << "\n0.\tId: 0\tName: loser";			

5	Assert getReselts() retuns the same string as test contains		true	true	
Post condition(s) for Test: getReselts() retuns the same string as test contains					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_getResults_03**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getResult() function in the STVElection class to make sure that it returns a correct string

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

test_stv_election_getResults.cc
STVElection(...)
addWinner()
addLoser()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballots objects, Shuffle = false			
2	Setup: Call addWinner()	new Candidate(0, "winner")			

3	Setup:
	Set test string stream
		this->test << "\nWinners vector:";
		this->test << "\n0.\tId: 0\tName: winner";
		this->test << "\nLosers vector:";
		this->test << "\nWinners in Order: ";
		this->test << "\n0.\tId: 0\tName: winner";
		this->test << "\nLosers			

		in Order: ";			
4	Assert getReselts() retuns the same string as test contains		true	true	

Post condition(s) for Test:
getReselts() retuns the same string as test contains

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----_ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_getResults_04**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the getResult() function in the STVElection class to make sure that it returns a correct string

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

test_stv_election_getResults.cc
STVElection(...)
addWinner()
addLoser()

Automated: Yes ----_ No __

Results: Pass ----_ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 1, cands = empty vector of Candidate objects, bals = empty vector of Ballots objects, Shuffle = false			
2	Setup: Call addLoser()	new Candidate(0, "loser")			

3	Setup:
	Set test string stream
		this->test << "\nWinners vector:";
		this->test << "\nLosers vector:";
		this->test << "\n0.\tId: 0\tName: loser";
		this->test << "\nWinners in Order: ";
		this->test << "\n0.\tId: 0\tName: loser";
		this->test << "\nLosers			

		in Order: ";			
4	Assert getReselts() retuns the same string as test contains		true	true	
Post condition(s) for Test: getReselts() retuns the same string as test contains					

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_removeLastLoser_01**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the removeLastLoser() function in the STVElection class to make sure that it returns the last candidate in losers_, if losers_is empty, then returns an candidate with id = -1 and name = “none”

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

test_stv_election_removeLastLoser.cc
STVElection(...)

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballots object, Shuffle = false			
2	Assert removeLastLoser() returns a candidate with id = -1		true	true	

Post condition(s) for Test:
removeLastLoser() returns a candidate with id = -1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_stv_election_removeLastLoser_02

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the removeLastLoser() function in the STVElection class to make sure that it returns the last candidate in losers_, if losers_is empty, then returns an candidate with id = -1 and name = “none”

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

test_stv_election_removeLastLoser.cc
STVElection(...)
addLoser()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballots object, Shuffle = false			
2	Setup: Call addLoser()	new Candidate(0, "test")			
3	Assert removeLastLoser() returns a candidate with id = 0		true	true	

Post condition(s) for Test:
removeLastLoser() returns a candidate with id = 0

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_removeLastLoser_03**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the removeLastLoser() function in the STVElection class to make sure that it returns the last candidate in losers_, if losers_is empty, then returns an candidate with id = -1 and name = “none”

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

test_stv_election_removeLastLoser.cc
STVElection(...)
addLoser()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballots object, Shuffle = false			
2	Setup: Call addLoser() for 100 times with integer i	for (int i = 0; i < 100; i++) addLoser(new Candidate(i, "test"));			
3	Assert removeLastLoser() returns a candidate with id = 99		true	true	

Post condition(s) for Test:
removeLastLoser() returns a candidate with id = 99

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 1 April 2020

Test Case ID#: UT_stv_election_shuffleBallotsr_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the shuffleBallots() function in the STVElection class to make sure that it shuffles the shuffledBallots_ vector randomly

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

test_stv_election_shuffleBallots.cc
STVElection(...)
is_not_same(std::vector<int> v1, std::vector<int>v2)
getShuffledBallots()

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = vector of 100 Ballots objects, Shuffle = false			
2	Call shuffleBallots()				
3	Set v1	getShuffledBallots()			
4	Call shuffleBallots()				
5	Set v2	getShuffledBallots()			
6	Assert is_not_same(v1, v2) returns true		true	true	

Post condition(s) for Test:
is_not_same(v1, v2) returns true which means v1 and v2 are different vectors

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: UT_stv_election_shuffleBallotsr_02

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the shuffleBallots() function in the STVElection class to make sure that it shuffles the shuffledBallots_ vector randomly

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

test_stv_election_shuffleBallots.cc
STVElection(...)
is_not_same(std::vector<int> v1, std::vector<int>v2)
getShuffledBallots()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	setup: Create a STVElection object	type = empty string, seats = 5, cands = empty vector of Candidate objects, bals = empty vector of Ballots objects, Shuffle = false			
2	Assert getShuffledBallots() returns empty vector of integers		true	true	

Post condition(s) for Test:
getShuffledBallots() returns empty vector of integers

VotingApp Class Unit Test Logs

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----_ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForFiles_001

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is behaving correctly by inputting the expected argument of ‘q’ to terminate the number of files.

This is a Good Data Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: Yes ----_ No __

Results: Pass ----_ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForFiles()	none	Program will not ask for file names.	Program did not ask for file names	If “q” is typed before any files are entered. The member variable of files_ will not be set.

Post condition(s) for Test:
files_ member variable is not set.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----_ System __ **Test Date:** 30 March 2020

Test Case ID#: **UT_votingapp_askForFiles_002**

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is robust and behaves properly when an invalid argument is input to terminate the input of files. Input ‘v’ as an attempt to terminate.

This is a Bad Data Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForFiles()	none	Program will ask for file names.	The program continued to ask for file names.	The program is not checking for validity of file names in this method. It will accept any entry.

Post condition(s) for Test:
files_ member variable is not set.

Project Name: Project 1: Voting System

Team# 4

Test Stage: [✓ Unit __] System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_askForFiles_003

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is behaving correctly by inputting a valid file name.

This is a Good Data Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: [✓ Yes __] No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForFiles()	none	files_ member variable is a vector<std::string> with file name.	Asks for another file because	

Post condition(s) for Test:
files_ member variable is set.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForFiles_004

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is behaving correctly by inputting two valid file names.

This is a Good Data Test and Boundary Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForFiles()	none	files_ member variable is a vector of size 2 with both file names.	Accepts both file names	

Post condition(s) for Test:
files_ member variable is set.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForFiles_005

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is behaving correctly by inputting three valid file names.

This is a Good Data Test and Boundary Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForFiles()	none	files_ member variable is a vector of size 3 with both file names.	Accpets all file names and the files_ size is now 3	

Post condition(s) for Test:
files_ member variable is set.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit _ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForFiles_006

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Test that the askForFiles() method is behaving correctly by inputting 10 valid file names.

This is a Boundary Data Test and Good Data Test

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

Test_votingapp_askFor.cc
askForFiles()

Automated: Yes ----__ No __

Results: Pass ___ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForFiles()	none	files_ member variable is a vector of size 10 with both file names.		

Post condition(s) for Test:
files_ member variable is set.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----__ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_doesFileExist_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Testing the doesFileExist() helper method for Good data when a file that exists is input.

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

Test_votingapp_doesFileExist.cc
doesFileExist()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
doesFileExist(std::string) has been called with an input string

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string file1 = “test1.csv”				Create string of a file name that exists in the local directory
2	Call doesFileExist(std:string), return to bool variable test1	none	Return true	Returned true	
3	assertm(test1, "File does exist");	none	none	none	

Post condition(s) for Test:
‘True’ bool is returned.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_doesFileExist_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Testing the doesFileExist() helper method for Bad data when a file that doesn’t exists is input.

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

Test_votingapp_doesFileExist.cc
doesFileExist()

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
doesFileExist(std::string) has been called with an input string

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string file2 = “2test.csv”				Create string of a file name that doesn’t exist in the local directory
2	Call doesFileExist(std:string) = test2	none	Return false	Returned false	
3	assertm(!test2, "File doesn’t exist");		none	none	

Post condition(s) for Test:
False bool is returned.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_doesFileExist_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Testing the doesFileExist() helper method for Bad data when an empty string is the input.

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

Test_votingapp_doesFileExist.cc
doesFileExist()

Automated: Yes [✓ __] No __

Results: Pass[✓ __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
doesFileExist(std::string) has been called with an empty string

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string file3 = “”				

2	Call doesFileExist(std:string)= test3	none	False bool returned	False was returned	
3	assertm(!test3, "string empty");		none	none	

Post condition(s) for Test:
False bool is returned

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
Good data test for askForSeats() method.
Will test when input value is 5.

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

Test_votingapp_askFor.cc
askForSeats()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForSeats()	none	5 returned	Returned value is 5	

Post condition(s) for Test:
Return value is 5

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a boundary test for askForSeats(). Input will be 10.

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

Test_votingapp_askFor.cc
askForSeats()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForSeats()	none	Return value is 10.	Return value is 10	

Post condition(s) for Test:
Return value is 10.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----__ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a boundary test and Bad data test for askForSeats(). Input will be 0.

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

Test_votingapp_askFor.cc
askForSeats()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

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

1	Call askForSeats()	none	User is prompted with a different input value.	No value returned, user is prompted to enter a valid input	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_04

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a Bad data test for askForSeats().
The input value for seats will be -1.

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

Test_votingapp_askFor.cc
askForSeats()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForSeats()	none	User is prompted with a different input value.	User is prompted to input a valid seat number.	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_05

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a boundary test for askForSeats().
The input value will be 1.

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

Test_votingapp_askFor.cc
askForSeats()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForSeats()	none	Return value of 1.	Return value is 1	

Post condition(s) for Test:
Return value of 1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForSeats_06

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForSeats().
The input value will be 11.

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

Test_votingapp_askFor.c
askForSeats()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForSeats()	none	User will be prompted to input another value.	User is prompted to input a valid seat number.	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForType_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForType(). The input will be "STV."

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

Test_votingapp_askFor.cc
askForType()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForType()	none	Return string of "STV"	Allowed input string and returned string of "STV"	

Post condition(s) for Test:
Return string of "STV"

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForType_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForType(). The input string will be “Plurality.”

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

Test_votingapp_askFor.cc
askForType()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForType()	none	Return string of “Plurality”	Accepted input string and returned a string of “Plurality”	

Post condition(s) for Test:
Return string of “Plurality.”

Project Name: Project 1: Voting System

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_askForType_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForSeats(). The input value will be “stv”

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

Test_votingapp_askFor.cc
askForType()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public

A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askFortType()	none	User prompted to input a valid string.	Did not accept user input. Prompted for valid input string.	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForType_04

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForType(). The input value will be "plurality."

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

Test_votingapp_askFor.cc
askForType()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForType()	none	User prompted for another input.	Invalid input string. Prompted user to input a valid string.	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForType_05

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForType(). The input will not be a 1.

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

Test_votingapp_askFor.cc
askForType()

Automated: [✓ Yes] __   No __

Results: [✓ Pass] __   Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForType()	none	User prompted for another input.	Prompted user for a valid input string	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System   **Team#** 4

Test Stage: [✓ Unit] __   System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_askForShuffleTurnOff_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForShuffleTurnOff(). The input will be "true."

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

N/A
askForShuffleTurnOff()

Automated: [✓ Yes] __   No __

Results: [✓ Pass] __   Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForShuffleTurnOff()	none	Return true.	Valid input. Returned true.	

Post condition(s) for Test:
Return true

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForShuffleTurnOff_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForShuffleTurnOff(). The input will be false.

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

Test_votingapp_askFor.cc
askForShuffleTurnOff()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForShuffleTurnOff()	none	Return false	Valid input. Return value was false	

Post condition(s) for Test:
Return false.

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForShuffleTurnOff_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForShuffleTurnOff(). The input will be "True."

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

Test_votingapp_askFor.cc
askForShuffleTurnOff()

Automated: Yes ---- __ No __

Results: Pass ---- __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call askForShuffleTurnOff()	none	Sorry, your input must equal 'true' or 'false.'	Input invalid. Prompts user for valid input.	

Post condition(s) for Test:
Return value of 1

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ---- __ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayelectionparam_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayElectionParams(int, string, bool).
Int = 1
String= "STV"
Bool = true

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

Test_votingapp_displayelectionparams.cc
displayElectionParams(int,string, bool)

Automated: Yes ---- __ No __

Results: Pass ---- __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayElectionParams()	none	Values will be printed out	Correct values printed out.	

Post condition(s) for Test:
VotingApp methods must be public
Parameters are displayed on the screen

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_displayelectionparams_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayElectionParams(int, string, bool).
Int = 10
String= “Plurality”
Bool = false

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

Test_votingapp_displayelectionparams.cc
displayElectionParams()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayElectionParams()	none	Values will be printed out	Correct input values are printed out	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_displayelectionparams_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for displayElectionParams(int, string, bool).
Int =
String= 1
Bool = false

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

Test_votingapp_displayelectionparams.cc
displayElectionParams(int, string, bool).

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call displayElectionParams(int, string, bool).	none	Error Message		

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForConfirmation_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForConfirmation(). The input will be “true.”

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

Test_votingapp_askFor.cc
askForConfirmation()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForConfirmation	none	Return true	Valid input argument. Return value is true	

Post condition(s) for Test:
True returned

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForConfirmation_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for askForConfirmation(). The input will be "false."

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

Test_votingapp_askFor.cc
askForConfirmation()

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForConfirmation()	none	Return false	Correct input value, return false	

Post condition(s) for Test:
False returned

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_askForConfirmation_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for askForConfirmation(). The input is "False."

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

Test_votingapp_askFor.cc
askForConfirmation()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call askForConfirmation()	none	Prompted to type either 'true' or 'false'	Invalid input. Prompt user for a valid input string.	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_processFiles_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for processFiles() and a boundary test. This will be a file with 3 candidates of unique names and 3 ballots.

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

processFiles()
test1.csv

Automated: [✓ Yes] __ No __

Results: Pass ---_ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
The files_ member variable has been set by the askForFiles() method.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string file = “test1.csv”		none	none	
2	Set files_ member variable to file name		none	none	
3	assertm(files_.size() == 1, “file push_back”)		none	none	
4	Call processFiles()	Test1.csv	Private member variables are set. Ballot vector size will be 3 and candidate vector size will be 3	Ballot size of 3 and candidate size of 3	
5	assertm(ballots_.size() == 3, “test1 ballots_”)		none	none	
6	assertm(candidates_.size() == 3, “test1 candidates_”);		none	none	
7	Clear files member variable		none	none	
8	assertm(votingapp.files_.empty(), “files empty”);		none	none	

Post condition(s) for Test:
3 Candidate Objects have been created
3 Ballot Objects have been created
Candidate member variable set
Ballot member variable set

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_processFiles_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test of a file with 1 candidates and 1 ballots for processFiles().

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

Test2.csv
processFiles()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
The files_ member variable has been set by the askForFiles() method.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Std::string file = “test2.csv”		none	none	
2	Set files_ member variable to file name		none	none	

3	assertm(files_.size() == 1, “file push_back”)		No message printed	No message	Would print “file push_back” if failed assert
4	Call processFiles()	Test2.csv	No error	No error	
5	assertm(ballots_.size() == 1, “test2 ballots_”)		No message printed	No message	Would print “test2 ballots_” if failed assert
6	assertm(candidates_.size() == 1, “test2 candidates_”);		No message printed	No message	Would print “test2 candidates_” if failed assert
7	Clear files member variable		none	none	
8	assertm(votingapp.files_.empty(), “files empty”);		No message printed	No message	Would print “files empty” if failed assert

Post condition(s) for Test:
1 Candidate Objects have been created
1 Ballot Objects have been created
Candidate member variable set
Ballot member variable set

Project Name: Project 1: Voting System

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#: **UT_votingapp_processFiles_03**

Name(s) of Testers: Sara Nelson (nels8907)

Test Description: This is a good data test and boundary test with 10 candidates and 100,000 ballots.

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

Test3.csv
processFiles()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
The files_ member variable has been set by the askForFiles() method.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string file = “test3.csv”		none	none	
2	Set files_ member variable to file name		none	none	
3	assertm(files_.size() == 1, “file push_back”)		No print message	No message	Would print “file push_back” if failed assert
4	Call processFiles()	Test3.csv	No error	No error	

5	assertm(ballots_.size() == 100000, “test3 ballots_”)		No print message	No message	Would print “test3 ballots_” if failed assert
6	assertm(candidates_.size() == 10, “test3 candidates_”);		No print message	No message	Would print “test3 candidates_” if failed assert
7	Clear files member variable		No error	No error	
8	assertm(votingapp.files_.empty(), “files empty”);		No print message	No message	Would print “files empty” if failed assert

Post condition(s) for Test:
10 Candidate Objects have been created
100000 Ballot Objects have been created
Candidate member variable set
Ballot member variable set

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----_ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_processFiles_04

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for processFiles() and a boundary test. Two separate input files of data.

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

processFiles()
Test1.csv
test4.csv

Automated: Yes ----_ No __

Results: Pass ----_ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
The files_ member variable has been set by the askForFiles() method.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Std::string file = “test1.csv” Std::string file2 = “test4.csv”		No error	No error	
2	Set files_ member variable to file name		none	none	
3	assertm(files_.size() == 2, “file push_back”)		No message	No message	Would print “file push_back” if failed
4	Call processFiles()	Test1.csv test4.csv	none	none	
5	assertm(ballots_.size() == 8, “test4 ballots_”)		none	none	Would print “test4 ballots_” if failed
6	assertm(candidates_.size() == 3, “test4 candidates_”);		none	none	Would print “test4 candidates_” if failed
7	Clear files member variable		none	none	

8	assertm(votingapp.files_.empty(), “files empty”);		none	none	Would print “files empty” if failed

Post condition(s) for Test:
3 Candidate Objects have been created
10 Ballot Objects have been created
Candidate member variable set
Ballot member variable set

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#: **UT_votingapp_displayHelp_01**

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayHelp(). Call the method to verify that it will return the help window.

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

Test_votingapp_displayHelp..cc
displayHelp();

Automated: Yes __ No __

Results: Pass ___ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayHelp();	none	Help text to be displayed	Text for help window is displayed	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_createCandidate_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for a string of one name and testing createCandidates();
Std::string “Mary”
Int = 1;

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

test_votingapp_createCandidate.cc
VotingApp:: vector<Candidate*>createCandidate(int i; std::string candidate);

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
Std::string candidates = “Mary”
Std::vector <Candidate> vCand;
Std:string name;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string candidates = “Mary” Std::vector <Candidates*> vCand Std::string name;		No error	No error	
2	vCand = votingapp.createCandidate (i, candidates);	none	vCand will be a vector of size 1 with name “Mary”	vCand was a vector of size 1 with value “Mary”	
3	assertm(vCand.size()==1 “candidate vector size”)		No message printed	No message printed	
4	assertm(vCand[0]->getName() ==”Mary”, “vCand name”		No message printed	No message printed	

Post condition(s) for Test:
Candidates_ member variable is set

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit ----__ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_createCandidate_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for a string of two names and testing createCandidates();
Std::string “Mary, John”
Int = 1;

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

test_votingapp_createCandidate.cc
VotingApp:: vector<Candidate*>createCandidate(int i; std::string candidate);

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
Std::string candidates = “Mary, John”
Std::vector <Candidate> vCand;
Std:string name;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string candidates = “Mary, John” Std::vector <Candidates*> vCand Std::string name;		None	None	
2	vCand = votingapp.createCandidate (i, candidates);	none	none	none	
3	assertm(vCand.size()==2 “candidate vector size”)		No message	No message	
4	assertm(vCand[0]->getName() == “Mary”, “Filed string name vCand[0]”);		No message	No message	

5	assertm(vCand[1]->getName() == “John”, “Filed string name vCand[1]”);		No message	No message	

Post condition(s) for Test:
Candidate_ member variable is set

Project Name: Project 1: Voting System

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#: **UT_votingapp_createCandidate_03**

Name(s) of Testers: Sara Nelson (nels8907)

Test Description:
This is a good data test and boundary test for a string of two names and testing createCandidates();
Std::string “Mary, John”
Int = 1;

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

test_votingapp_createCandidate.cc
VotingApp:: vector<Candidate*>createCandidate(int i; std::string candidate);

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
Std::string candidates = "Mary, John, Susan, Karen, Cathy, Debbie, Collin, Dugg, Jack, Pat";
Std::vector <Candidate> vCand;
Std:string name;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Std::string candidates = “Mary, John”		None	None	

	Std::vector <Candidates*> vCand Std::string name;				
2	Call processFiles()	none	No error	No error	
3	assertm(vCand.size()==10 “candidate vector size”)		No message	No message	
4	assertm(vCand[0]->getName() == “Mary”, “Filed string name vCand[0]”);		No message	No message	
5	assertm(vCand[1]->getName() == “John”, “Filed string name vCand[1]”);		No message	No message	
6	assertm(vCand[2]->getName() == “Susan”, “Filed string name vCand[2]”);		No message	No message	
7	assertm(vCand[3]->getName() == “Karen”, “Filed string name vCand[3]”);		No message	No message	
8	assertm(vCand[4]->getName() == “Cathy”,		No message	No message	

	“Filed string name vCand[4]”);				
9	assertm(vCand[5]->getName() == “Debbie”, “Filed string name vCand[5]”);		No message	No message	
10	assertm(vCand[6]->getName() == “Collin”, “Filed string name vCand[6]”);		No message	No message	
11	assertm(vCand[7]->getName() == “Dugg”, “Filed string name vCand[7]”);		No message	No message	
12	assertm(vCand[8]->getName() == “Jack”, “Filed string name vCand[8]”);		No message	No message	
13	assertm(vCand[9]->getName() == “Pat”, “Filed string name vCand[9]”);		No message	No message	

Post condition(s) for Test:
Candidates_ member variable was set

Test Stage: Unit ----__ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayauditFileLocation_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayAuditFileLocation(); when a STV Election is run

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

Test_votingapp_displayauditlocation.cc
displayAuditFileLocation()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
VotingApp method must be public
Election methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
A STV Election object must be created.
Audit File location must be set by setAuditFileLocation(“”);
Location string == “STV File Path

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create and set variable std:string file = “test1.csv”		No error	No error	
2	Set member variable		None	None	
3	Call processFiles() //tested earlier		None	None	
4	Create new STV Election with input parameters				Input parameters are not important. But they need to match what STVElection is expecting

5	Call setAuditFilePath(“STV File Path”)		No error	No error	
6	Call displayAuditFileLocation();	none	The audit file path will be printed.	“STV File Path” was printed	
7	Clear files_ member variable		none	none	
8	assertm(votinapp.files_.emptu(), “clear files_”)		No message	No message	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_displayauditFileLocation_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayAuditFileLocation(); when a Plurality Election is run

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

Test_votingapp_displayauditlocation.cc
displayAuditFileLocation()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.
A Plurality Election object must be created.
Audit File location must be set by setAuditFileLocation(“”);
Location string == “Plurality File Path”

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

1	Create and set variable std:string file = “test1.csv”		No error	No error	
2	Set member variable		None	None	
3	Call processFiles() //tested earlier		None	None	
4	Create new PluralityElection with input parameters				Input parameters are not important. But they need to match whatPluralityElection is expecting
5	Call setAuditFilePath(“Plurality File Path”)		No error	No error	
6	Call displayAuditFileLocation( );	none	The audit file path will be printed.	“Plurality File Path” was printed	
7	Clear files_ member variable		none	none	
8	assertm(votinapp.files_.emptu(), “clear files_”)		No message	No message	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System

Team# 4

Test Stage: Unit _
System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayerror_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for when an error message is called with a valid input string.

Std::string = “error”

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

Test_votingapp_displayerror.cc
displayError(std::string);

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayError(“error”)	none	Error will be printed out	“Error” was printed	

Post condition(s) for Test:
N/A

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayerror_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for when an error message is called with a valid input string.

Std::string = “”

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

Test_votingapp_displayerror.cc
displayError(std::string);

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayError(“”)	none	Empty string will be printed	Nothing was printed due to the empty string	

Post condition(s) for Test:
N/A

Team# 4

Test Stage: Unit _ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayerror_03

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for when an error message is called with a valid input string.

Std::string = “”.

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

Test_votingapp_displayerror.cc
displayError(std::string);

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call displayError(“This is an error message”)	none	Error will be printed.	“This is an error message” was printed	

Post condition(s) for Test:
N/A

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit [✓ __] System __ **Test Date:** 30 March 2020

Test Case ID#:
UT_votingapp_displayResults_01

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayResults when a STV Election is run.

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

Test_votingapp_displayResults.cc
displayResults()

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create and set variable std:string file = “test1.csv”		None	None	
	Set member variable		None	None	
	Call processFiles()		None	NOne	
	Create new STVElection with input parameters		None	None	

1	assertm(!votingapp.election_.->getResults().empty(), “No results”);	none	No Message	No Message	
	votingapp.displayResults()		The output will display the winners in order and the losers in order.	The output displays the test for winners and losers. Indicated that getResults() was properly called.	The getResults() method from election will not display much unless runAlgorithm() is executed
	Clear files_ member variable		none	none	
	assertm(votingapp.files_.empty(), “clear files_”		No message	No message	

Post condition(s) for Test:
N/A

Team# 4

Test Stage: [✓ Unit] __ System __

Test Date: 30 March 2020

Test Case ID#:
UT_votingapp_displayResults_02

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for displayResults when a Plurality Election is run.

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

Test_votingapp_displayResults.cc
displayResults()

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
VotingApp methods must be public
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create and set variable std:string file = “test1.csv”		None	None	
	Set member variable		None	None	
	Call processFiles()		None	None	
	Create new PluralityElection with input parameters		None	None	
1	assertm(!votingapp.election_.->getResults().empty(), “No results”);	none	No Message	No Message	
	votingapp.displayResults()		The output will display the winners in order and the losers in order.	The output displays the test for winners and losers. Indicated that getResults() was properly called.	The getResults() method from election will not display much unless runAlgorithm() is executed
	Clear files_ member variable		None	None	
	assertm(votingapp.files_.empty(), “clear files_”		No message	No message	

Team# 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#: UT_votingapp_run_01 **Name(s) of Testers:**
Sara Nelson (nels8907)

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test1.csv
q
1
Plurality
true

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

votingapp_run_01.txt
run()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_1.txt	votingapp_run_1.txt	The method will have to go through all the prompts without asking again	See results	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: Unit __ System __ **Test Date:** 30 March 2020

Test Case ID#: UT_votingapp_run_02

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test1.csv
q
1
STV
true

Name(s) of Testers:
Sara Nelson (nels8907)

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

votingapp_run_02.txt
run()

Automated: Yes ----__ No __

Results: Pass ----__ Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_2.txt	votingapp _run_2.txt	The method will go through all the prompts without having to repeat a user input	See results	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: [✓ Unit] System __

Test Date: 30 March 2020

Test Case ID#: **UT_votingapp_run_03**

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test1.csv
test2.csv

test4.csv
quit.csv
q
3
STV
true

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

votingapp_run_03.txt
run()

Automated: [✓ Yes] No __

Results: [✓ Pass] Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_3.txt	votingapp _run_3.txt	The method will go through all the prompts with having to repeat a question after test2.csv	See results	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set

helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#: UT_votingapp_run_04

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a bad data test for the run() method in votingapp.
Inputs:
test1.csv
test2.csv
test4.csv
test5.csv
test6.csv
test7.csv
q
2
STV
false
true

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

votingapp_run_04.txt
run()

Automated: Yes __ No __

Results: Pass l__ Fail__

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

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

1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_4.txt	votingapp _run_4.txt	Endless loop	Endless loop	Since the inputs are a text file instead of manual by user, this creates an endless loop

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: Unit __ System __

Test Date: 30 March 2020

Test Case ID#: **UT_votingapp_run_05**

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test2.csv
test4.csv
test5.csv
test6.csv
test7.csv
q
2
STV
false
test2.csv
test4.csv
test5.csv

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

votingapp_run_05.txt
run()

test6.csv
test7.csv
q
2
STV
true

Automated: Yes [✓ __] No __

Results: Pass [✓ __] Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_5.txt	votingapp _run_5.txt	After the user denies confirmation, they will be repromted to input everything again	See results	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: Unit [✓ __] System __ **Test Date:** 30 March 2020

Test Case ID#: **UT_votingapp_run_06**

Name(s) of Testers:
Sara Nelson (nels8907)

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test1.csv
test2.csv
test4.csv
test5.csv
q
1
Plurality
true

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

votingapp_run_06.txt
run()

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_6.txt	votingapp _run_6.txt	User will input all the files and then be prompted with each question	See results section	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Team# 4

Test Stage: Unit _ System __

Test Case ID#: **UT_votingapp_run_07**

Test Description:
This is a good data test for the run() method in votingapp.
Inputs:
test2.csv
test4.csv
test5.csv
test6.csv
q
2
STV
false
test2.csv
test4.csv

test5.csv
test6.csv

q
2

STV
true

Test Date: 30 March 2020

Name(s) of Testers:
Sara Nelson (nels8907)

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

votingapp_run_07.txt
run()

Automated: Yes _ No __

Results: Pass __ Fail___

Preconditions for Test:
A VotingApp object has been created using the class constructor with valid test_mode parameters.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Make program		Program compiles, no error		
2	./voting_app > votingpp_run_7.txt	votingapp _run_7.txt	User will be prompted for each method call. When confirmation is denied, user will be reprompted to type each input again.	See results	Run program with designated inputs from the text file

Post condition(s) for Test:
Candidates_ member variable is set
Files_ member variable is set
Ballots_ member variable is set
helpMessage_ member variable is set
Test_ member variable is set
*election_ member variable is set

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __

Test Date: 1 April 2020

Test Case ID#: UT_voting_app_is_number_01

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the is_number() function in the STVElection class to make sure that it returns correct value about whether a string is a non-negative integer

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

test_voting_app_is_number.cc

Automated: Yes _ No __

Results: Pass _ Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert is_number() returns true	"1234567891093874392847392"	true	true	

Post condition(s) for Test:
is_number() returns true

Project Name: Project 1: Voting System **Team#** 4

Test Stage: Unit __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_is_number_02**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the is_number() function in the STVElection class to make sure that it returns correct value about whether a string is a non-negative integer

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

test_voting_app_is_number.cc

Automated: Yes __ No __

Results: Pass __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert is_number() returns false	"1.1"	true	true	

Post condition(s) for Test:
is_number() returns false

Project Name: Project 1: Voting System

Team# 4

Test Stage: [✓ Unit] __ System __

Test Date: 1 April 2020

Test Case ID#: **UT_stv_election_is_number_03**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the is_number() function in the STVElection class to make sure that it returns correct value about whether a string is a non-negative integer

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

test_voting_app_is_number.cc

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Assert is_number() returns true	"0"	true	true	

Post condition(s) for Test:
is_number() returns true

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit __] System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_is_number_04**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the is_number(string) function in the STVElection class to make sure that it returns correct value about whether a string is a non-negative integer

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

test_voting_app_is_number.cc

Automated: [✓ Yes __] No __

Results: [✓ Pass __] Fail___

Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert is_number() returns false	"-1"	true	true	

Post condition(s) for Test:
is_number() returns false

Project Name: Project 1: Voting System **Team#** 4

Test Stage: [✓ Unit] __ System __ **Test Date:** 1 April 2020

Test Case ID#: **UT_stv_election_is_number_05**

Name(s) of Testers:
Yifan Zhang (zhan4372)

Test Description:
Test of the is_number(string) function in the STVElection class to make sure that it returns correct value about whether a string is a non-negative integer

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

test_voting_app_is_number.cc

Automated: [✓ Yes] __ No __

Results: [✓ Pass] __ Fail___

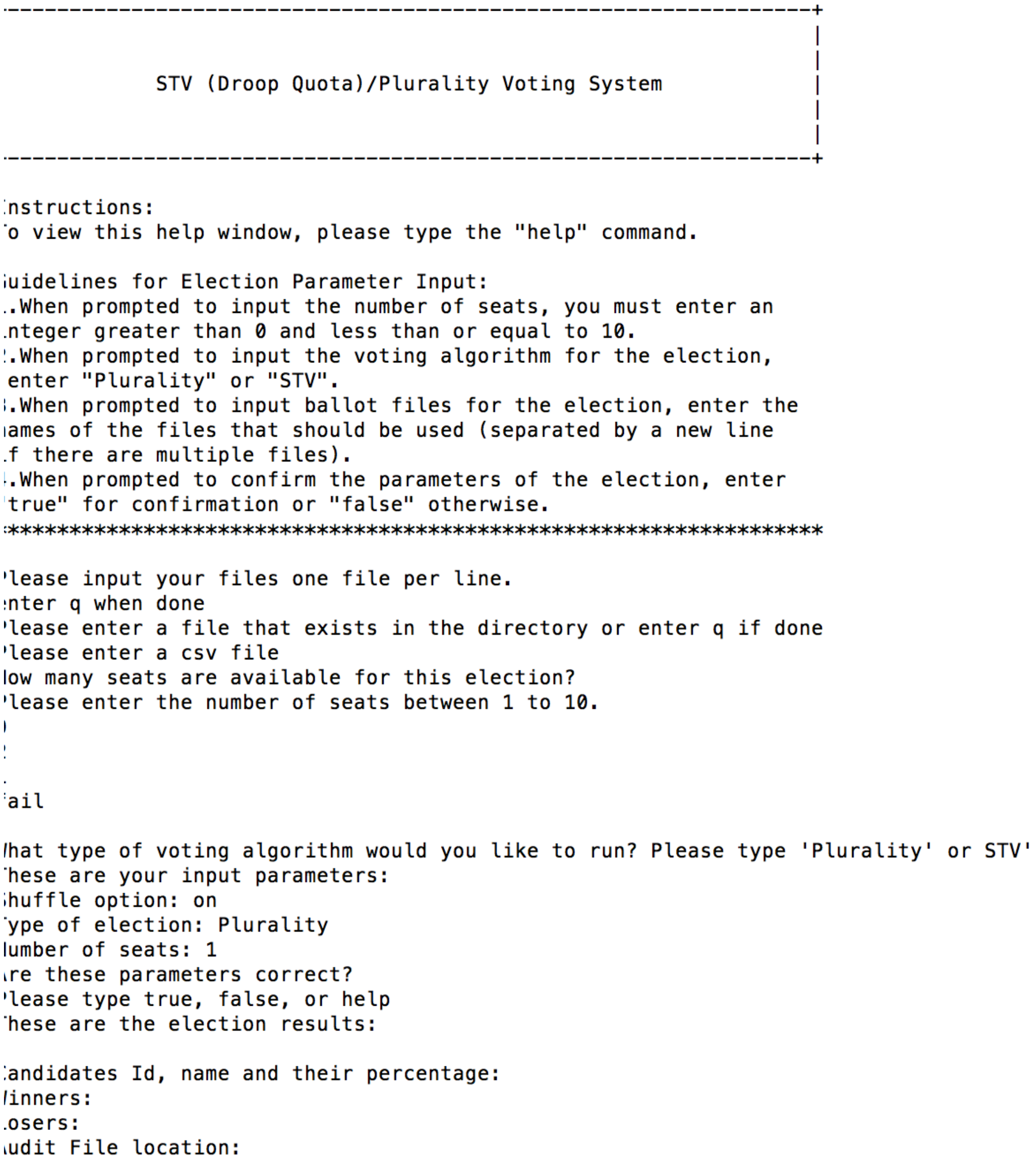
Preconditions for Test:
N/A

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Assert is_number() returns false	"98098asdjlakdnwljadb"	true	true	

Post condition(s) for Test:
is_number() returns false

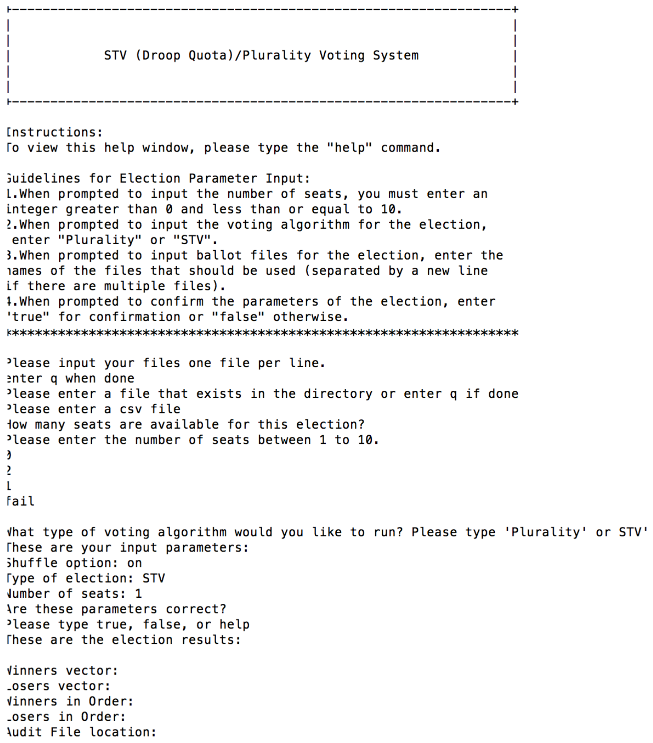
Results:

UT_votingapp_run_01



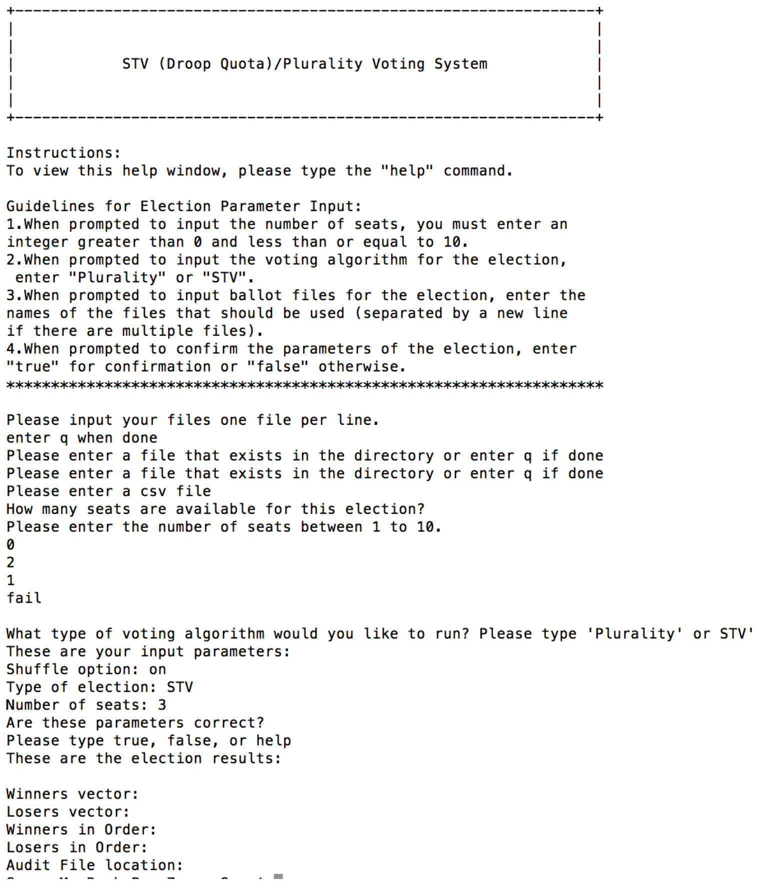
```
---------------------------------------------------------------+
                                                               |
                                                               |
            STV (Droop Quota)/Plurality Voting System          |
                                                               |
                                                               |
---------------------------------------------------------------+

nstructions:
'o view this help window, please type the "help" command.

iuidelines for Election Parameter Input:
..When prompted to input the number of seats, you must enter an
nteger greater than 0 and less than or equal to 10.
!.When prompted to input the voting algorithm for the election,
enter "Plurality" or "STV".
i.When prompted to input ballot files for the election, enter the
iames of the files that should be used (separated by a new line
.f there are multiple files).
i.When prompted to confirm the parameters of the election, enter
'true" for confirmation or "false" otherwise.
:****************************************************************

'lease input your files one file per line.
:nter q when done
'lease enter a file that exists in the directory or enter q if done
'lease enter a csv file
low many seats are available for this election?
'lease enter the number of seats between 1 to 10.
)
:
.
'ail

/hat type of voting algorithm would you like to run? Please type 'Plurality' or STV'
'hese are your input parameters:
ihuffle option: on
'ype of election: Plurality
lumber of seats: 1
ire these parameters correct?
'lease type true, false, or help
'hese are the election results:

:andidates Id, name and their percentage:
/inners:
.osers:
iudit File location:
```

UT_votingapp_run_02



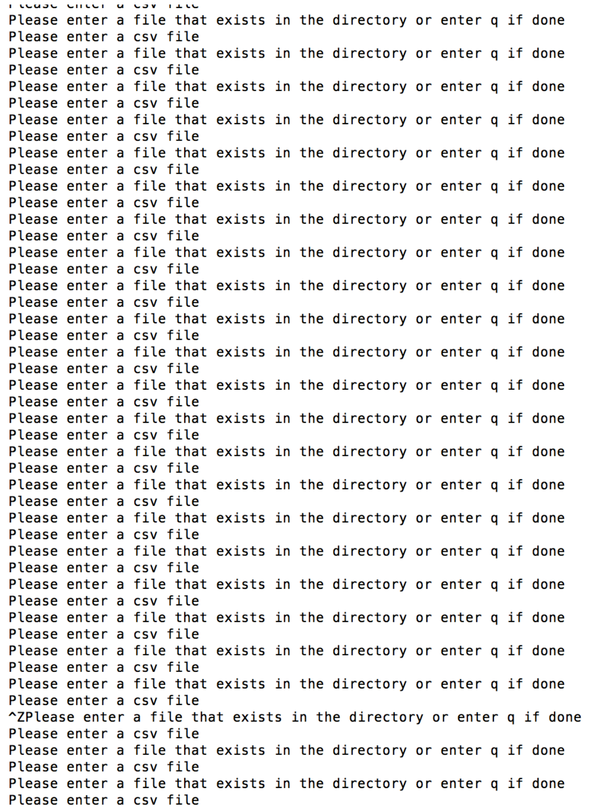
```
+------------------------------------------------------------------+
|                                                                  |
|                                                                  |
|            STV (Droop Quota)/Plurality Voting System             |
|                                                                  |
|                                                                  |
+------------------------------------------------------------------+

Instructions:
To view this help window, please type the "help" command.

Guidelines for Election Parameter Input:
1.When prompted to input the number of seats, you must enter an
integer greater than 0 and less than or equal to 10.
2.When prompted to input the voting algorithm for the election,
 enter "Plurality" or "STV".
3.When prompted to input ballot files for the election, enter the
names of the files that should be used (separated by a new line
if there are multiple files).
4.When prompted to confirm the parameters of the election, enter
"true" for confirmation or "false" otherwise.
******************************************************************

Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 1
Are these parameters correct?
Please type true, false, or help
These are the election results:

Winners vector:
Losers vector:
Winners in Order:
Losers in Order:
Audit File location:
```

UT_votingapp_run_03



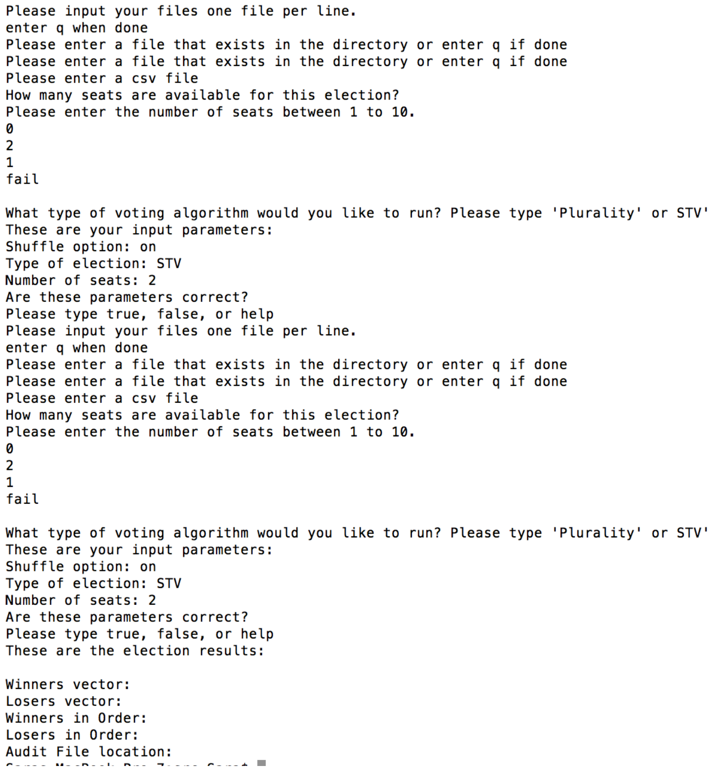
```
+-------------------------------------------------------------------+
|                                                                   |
|                                                                   |
|             STV (Droop Quota)/Plurality Voting System             |
|                                                                   |
|                                                                   |
+-------------------------------------------------------------------+

Instructions:
To view this help window, please type the "help" command.

Guidelines for Election Parameter Input:
1.When prompted to input the number of seats, you must enter an
integer greater than 0 and less than or equal to 10.
2.When prompted to input the voting algorithm for the election,
 enter "Plurality" or "STV".
3.When prompted to input ballot files for the election, enter the
names of the files that should be used (separated by a new line
if there are multiple files).
4.When prompted to confirm the parameters of the election, enter
"true" for confirmation or "false" otherwise.
*******************************************************************

Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 3
Are these parameters correct?
Please type true, false, or help
These are the election results:

Winners vector:
Losers vector:
Winners in Order:
Losers in Order:
Audit File location:
```

UT_votingapp_run_04



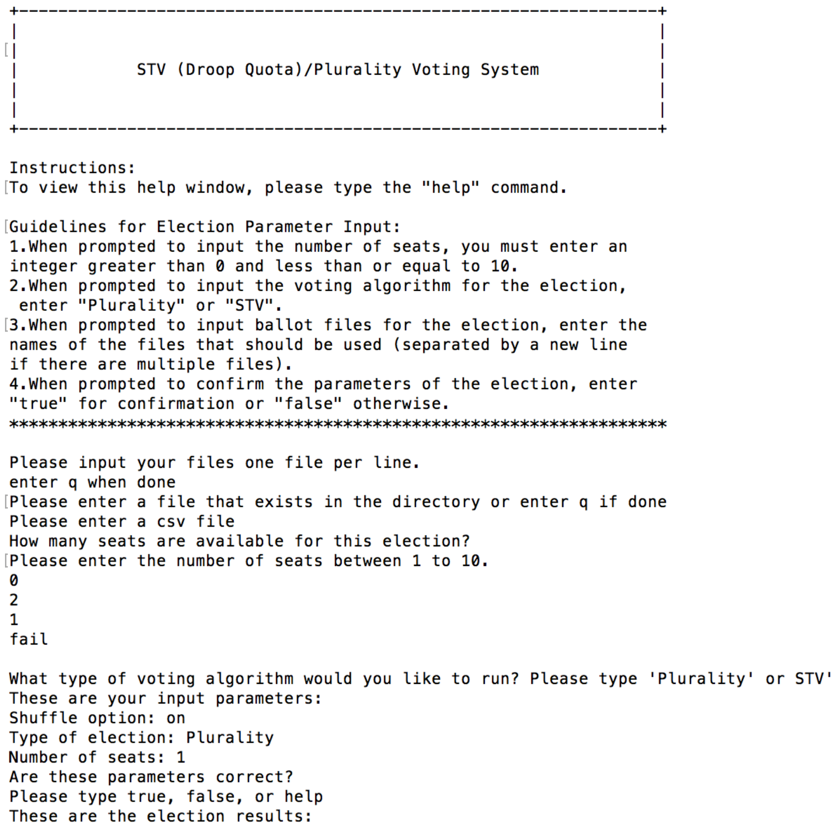
```
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
^ZPlease enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
```

UT_votingapp_run_05



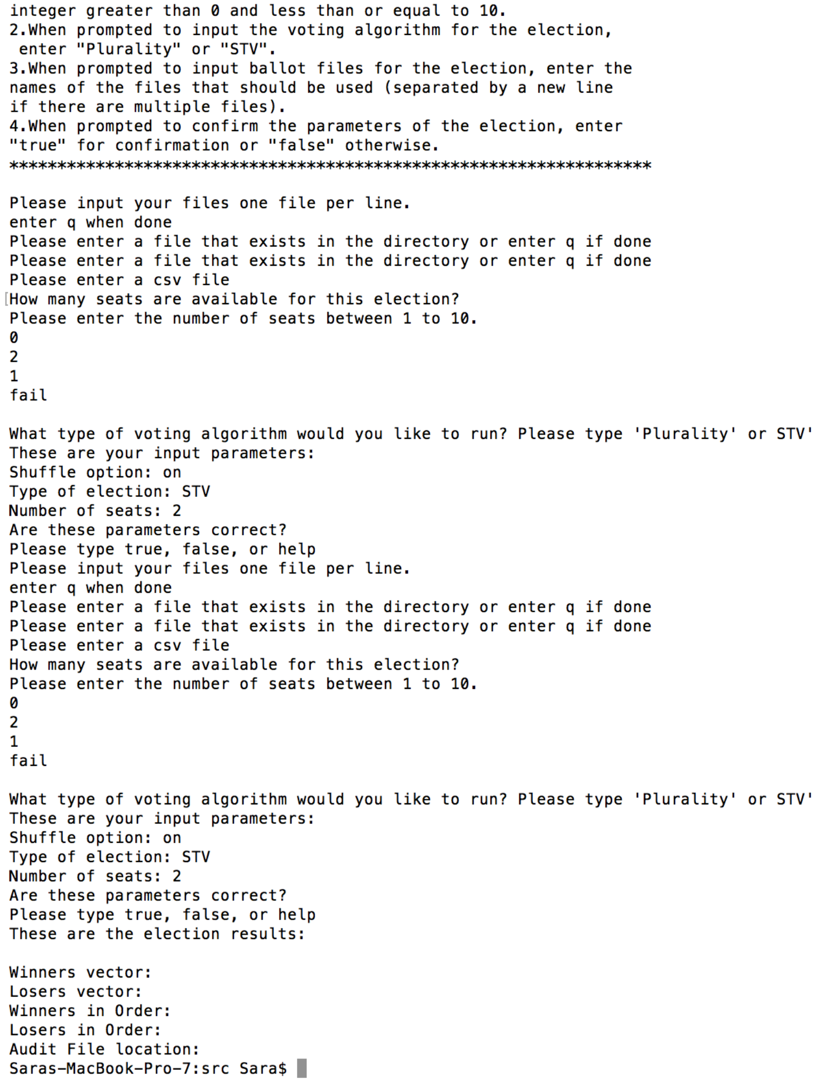
```
Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 2
Are these parameters correct?
Please type true, false, or help
Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 2
Are these parameters correct?
Please type true, false, or help
These are the election results:

Winners vector:
Losers vector:
Winners in Order:
Losers in Order:
Audit File location:
Saras-MacBook-Pro-7:src Sara$
```

UT_votingapp_run_06



```
+-----------------------------------------------------------------+
|                                                                 |
|                                                                 |
|            STV (Droop Quota)/Plurality Voting System            |
|                                                                 |
|                                                                 |
+-----------------------------------------------------------------+

Instructions:
To view this help window, please type the "help" command.

Guidelines for Election Parameter Input:
1.When prompted to input the number of seats, you must enter an
integer greater than 0 and less than or equal to 10.
2.When prompted to input the voting algorithm for the election,
 enter "Plurality" or "STV".
3.When prompted to input ballot files for the election, enter the
names of the files that should be used (separated by a new line
if there are multiple files).
4.When prompted to confirm the parameters of the election, enter
"true" for confirmation or "false" otherwise.
*******************************************************************

Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: Plurality
Number of seats: 1
Are these parameters correct?
Please type true, false, or help
These are the election results:
```

UT_votingapp_run_07



```
integer greater than 0 and less than or equal to 10.
2.When prompted to input the voting algorithm for the election,
 enter "Plurality" or "STV".
3.When prompted to input ballot files for the election, enter the
names of the files that should be used (separated by a new line
if there are multiple files).
4.When prompted to confirm the parameters of the election, enter
"true" for confirmation or "false" otherwise.
*******************************************************************

Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 2
Are these parameters correct?
Please type true, false, or help
Please input your files one file per line.
enter q when done
Please enter a file that exists in the directory or enter q if done
Please enter a file that exists in the directory or enter q if done
Please enter a csv file
How many seats are available for this election?
Please enter the number of seats between 1 to 10.
0
2
1
fail

What type of voting algorithm would you like to run? Please type 'Plurality' or STV'
These are your input parameters:
Shuffle option: on
Type of election: STV
Number of seats: 2
Are these parameters correct?
Please type true, false, or help
These are the election results:

Winners vector:
Losers vector:
Winners in Order:
Losers in Order:
Audit File location:
Saras-MacBook-Pro-7:src Sara$
```