Project 1: Voting System Team #10

—--

Test Stage: (Unit or system) Test Date:
Test Case ID: Name of Tester:
Test Description:
Test Storage Location and Functions Used:
Automated? (Yes or No)
Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					

Postconditions for Test:

—--
—--

Test Stage: Unit Test Date: 3/25/23
Test Case ID: BallotNode_getNext Name of Tester: Jacob
Test Description: Tests to ensure the getNext() function is working properly.
Test Storage Location and Functions Used: Found in BallotNode_unittest.cc and uses setIndex(), setNext(), getNext(), and getIndex() which are methods of the BallotNode class.
Automated? Yes
Preconditions for Test: Two BallotNodes are created, and one's next field is set to point to the other. Both have indexes set.

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

1	Checks to ensure that the node's next node has the correct index of 5.	Operates on two BallotNode objects which have been instantiated and linked.	The index of the node's next node should be 5.	Returns 5 correctly.	None
2	Reverses the connection so now the second node points to the first; checks to make sure the getNext() still works.	Works on the same two BallotNodes as before.	The index returned should be 3.	Returns 3 correctly.	None

Postconditions for Test: Both nodes which were instantiated dynamically are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: BallotNode_getIndex Name of Tester: Jacob
Test Description: Cheks to ensure the getIndex() method in BallotNode is working properly.
Test Storage Location and Functions Used: Found in BallotNode_unittest.cc and uses setIndex(), getIndex() methods in BallotNode.
Automated? Yes
Preconditions for Test: One BallotNode object is created and its index is set to 1.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The node created is checked to ensure	One BallotNode object.	Should return 1 as the index.	Returns 1 correctly.	None.

	the correct index is returned by getIndex().				
2	The nodes index is changed to 9999 and then getIndex() is used to get it.	The same BallotNode object.	Should return 9999 as the index.	Returns 9999 correctly.	None

Postconditions for Test: The created BallotNode is deleted.

—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: BallotNode_setIndex Name of Tester: Jacob
Test Description: Checks to ensure the setIndex() method in BallotNode is working properly.
Test Storage Location and Functions Used: Found in BallotNode_unittest.cc and uses setIndex() and getIndex() in BallotNode.
Automated? Yes
Preconditions for Test: One BallotNode object is created and its index is set to 7.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to make sure that the index is set to 7.	A single BallotNode object with index 7.	Should return 7.	Returns 7 correctly.	None
2	Checks to make sure that the index is -9000 after changing it.	The same BallotNode object as before.	Should return -9000.	Returns -9000 correctly.	None

Postconditions for Test: BallotNode that was created is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: BallotNode_setNext Name of Tester: Jacob

Test Description: Checks to ensure the setNext() method in BallotNode is working properly.

Test Storage Location and Functions Used: Found in BallotNode_unittest.cc and uses setIndex(), setNext(), getNext() methods in BallotNode.

Automated? Yes

Preconditions for Test: Two BallotNodes are created, one is set to point to the other. Both are given initial indices.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the BallotNode that the first BallotNode points to has index 500.	Operates on two BallotNodes, one with index 500 and the other with -100.	Should return 500.	Returns 500 correctly.	None.
2	Checks to make sure that the node that the second node points to has an index of -100 after calling setNext().	The same two BallotNode objects as before.	Should return -100.	Returns -100 correctly.	None.

Postconditions for Test: The two BallotNode objects that were created are deleted.

—---

—---

Test Stage: Unit
Test Date: 3/25/23
Test Case ID: BallotLinkedList_removeHead
Name of Tester: Jacob
Test Description: Checks to ensure that the removeHead() method in BallotLinkedList.cc works correctly.
Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses appendNode(), getLength(), getHead(), getTail(), removeHead() methods in BallotLinkedList and BallotNode.
Automated? Yes
Preconditions for Test: A BallotLinked list is created with three nodes with indices 1, 2, and 3.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks the length of the list is 2 after removing the head for the first time.	The BallotLinkedList with three BallotNodes in it.	Should return 2.	Returns 2 correctly.	None.
2	Checks the head of the list has value 2 after removing the head for the first time.	The same BallotLinkedList as before.	Should return 2.	Returns 2 correctly.	None.
3	Checks that the index of the tail is 3 after removing the head for the first time.	The same BallotLinkedList as before.	Should return 3.	Returns 3 correctly.	None.
4	Checks the length of the list is 1 after removing the head	The BallotLinkedList with three	Should return 1.	Returns 1 correctly.	None.

	for the second time.	BallotNodes in it.			
5	Checks the head of the list has value 3 after removing the head for the second time.	The same BallotLinkedList as before.	Should return 3.	Returns 3 correctly.	None.
6	Checks that the index of the tail is 3 after removing the head for the second time.	The same BallotLinkedList as before.	Should return 3.	Returns 3 correctly.	None.
7	Checks the length of the list is 0 after removing the heaf for the third time.	The BallotLinkedList with three BallotNodes in it.	Should return 0.	Returns 0 correctly.	None.
8	Checks the head of the list has value -1 after removing the head for the third time.	The same BallotLinkedList as before.	Should return -1.	Returns -1 correctly.	None.
9	Checks that the tail is nullptr after removing the head for the third time.	The same BallotLinkedList as before.	Should return nullptr.	Returns nullptr correctly.	None.
10	Checks that the length is unchanged from 0 when we try to remove the head	The same BallotLinkedList as before.	Should return 0.	Returns 0 correctly.	None.

	of an empty list.				

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.
—---
—---
Test Stage: Unit Test Date: 3/25/23
Test Case ID: BallotLinkedList_appendNode Name of Tester: Jacob
Test Description: Checks to ensure that the appendNode() method in BallotLinkedList.cc works correctly.
Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses appendNode(), getLength(), getHead(0, getTail(), getIndex() methods in BallotLinkedList and BallotNode.
Automated? Yes
Preconditions for Test: A BallotLinkedList is created, and one node is appended to it with index 1.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tests that the length of the list is 1 after appending 1 node.	The BallotLinkedList that was created.	Should return 1.	Returns 1 correctly.	None
2	Tests that the head of the list is 1 after appending 1 node.	The same BallotLinkedList.	Should return 1.	Returns 1 correctly.	None
3	Tests that the index of the tail is 1 after appending 1 node.	The same BallotLinkedList.	Should return 1.	Returns 1 correctly.	None
4	Tests that the length of the list is 2 after appending another	The same BallotLinkedList.	Should return 2.	Returns 2 correctly.	None

	node.				
5	Tests that the head of the list is 1 after appending a second node.	The same BallotLinkedList.	Should return 1.	Returns 1 correctly.	None
6	Tests that the index of the tail is now 2 after appending a second node.	The same BallotLinkedList.	Should return 2.	Returns 2 correctly.	None.

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: BallotLinkedList_getHead Name of Tester: Jacob

Test Description: Checks to ensure that the getHead() method in BallotLinkedList.cc works correctly.

Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses appendNode() and getHead() methods in BallotLinkedList.

Automated? Yes

Preconditions for Test: A BallotLinkedList is created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Ensures that the value of the head is -1 when the list is empty.	The BallotLinkedList created.	Should return -1.	Returns -1 correctly.	None.
2	Ensures that the	The	Should return 1.	Returns 1 correctly.	None.

	head has value 1 when we append a node to it.	BallotLinkedList created.			
3	Ensures that the head still has value 1 after appending a second node with value 2.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.
4	Ensures that the head now has value 2 after removing the old head.	The BallotLinkedList created.	Should return 2.	Returns 2 correctly.	None.
5	Ensures that the value of the head is -1 when no nodes are left.	The BallotLinkedList created.	Should return -1.	Returns -1 correctly.	None.

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: BallotLinkedList_populateList Name of Tester: Jacob

Test Description: Checks to ensure that the populateList() method in BallotLinkedList.cc works correctly.

Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses populateList(), getLength(), getHead(), getTail(), and cleanup() methods in BallotLinkedList.

Automated? Yes

Preconditions for Test: A BallotLinkedList is created. Three integer vectors with three numbers are created.

Step #	Test Step	Test Data	Expected Result	Actual Result	Notes

	Description				
1	Ensures that the list has length 3 after populating it.	The BallotLinkedList created.	Should return 3.	Returns 3 correctly.	None.
2	Ensures that the value of the head is 1.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.
3	Ensures that the index of the tail is 3.	The BallotLinkedList created.	Should return 3.	Returns 3 correctly.	None.
4	After cleaning and populating the list, checks that list length is 3.	The BallotLinkedList created.	Should return 3.	Returns 3 correctly.	None.
5	Checks to see if the head of the list has value 1.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.
6	Chekcs that the index of the tail is 2.	The BallotLinkedList created.	Should return 2.	Returns 2 correctly.	None.
7	Checks that the length is 0 after trying to populate list with all -1.	The BallotLinkedList created.	Should return 0.	Returns 0 correctly.	None.
8	Checks that the list has head value -1.	The BallotLinkedList	Should return -1.	Returns -1 correctly.	None.

		created.			
9	Checks that the tail is a nullptr.	The BallotLinkedList created.	Should return nullptr.	Returns nullptr correctly.	None.

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: BallotLinkedList_cleanup Name of Tester: Jacob

Test Description: Checks to ensure that the cleanup() method in BallotLinkedList.cc works correctly.

Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses appendNode(), cleanup(), getLength(), getHead(), and getTail() methods in BallotLinkedList.

Automated? Yes

Preconditions for Test: A BallotLinkedList is created. Two nodes are appended and then cleanup is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the length is 0 after cleanup.	The BallotLinkedList created.	Should return 0.	Returns 0 correctly.	None.
2	Checks that the head has index -1 after cleanup.	The BallotLinkedList created.	Should return -1.	Returns -1 correctly.	None.
3	Checks that the tail is a nullptr after cleanup.	The BallotLinkedList created.	Should return nullptr.	Returns nullptr correctly.	None.

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---

—---

Test Stage: Unit　　　　　　　　　　　　　　　　　　　　Test Date: 3/25/23

Test Case ID: BallotLinkedList_getLength　　　　　　　　　　Name of Tester: Jacob

Test Description: Checks to ensure that the getLength() method in BallotLinkedList.cc works correctly.

Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses appendNode(), removeHead(), and getLength() methods in BallotLinkedList.

Automated? Yes

Preconditions for Test: A BallotLinkedList is created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the list has length 0 before adding any nodes.	The BallotLinkedList created.	Should return 0.	Returns 0 correctly.	None.
2	Checks to ensure that the length is 1 after appending a node.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.
3	Checks that the length is 2 after appending another node.	The BallotLinkedList created.	Should return 2.	Returns 2 correctly.	None.
4	Checks that the length is 1 after removing a node.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: BallotLinkedList_getTail Name of Tester: Jacob
Test Description: Checks to ensure that the getTail() method in BallotLinkedList.cc works correctly.
Test Storage Location and Functions Used: Found in BallotLinkedList_unittest.cc and uses getTail(), appendNode(), and getIndex() methods in BallotLinkedList.
Automated? Yes
Preconditions for Test: A BallotLinkedList is created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the tail is a nullptr before adding any nodes.	The BallotLinkedList created.	Should return nullptr.	Returns nullptr correctly.	None.
2	Checks that the tail has index 1 after a node is appended.	The BallotLinkedList created.	Should return 1.	Returns 1 correctly.	None.
3	Checks that the tail has index 2 when another node is added.	The BallotLinkedList created.	Should return 2.	Returns 2 correctly.	None.
4	Checks that the tail has index 2 after removing the head node.	The BallotLinkedList created.	Should return 2.	Returns 2 correctly.	None.
5	Checks that the tail	The	Should return	Returns nullptr	None.

	is again nullptr when all nodes are removed.	BallotLinkedList created.	nullptr.	correctly.	

Postconditions for Test: The nodes in the list are destroyed and the list is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getAudit Name of Tester: Jacob

Test Description: Checks to ensure that the getAudit() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setAudit() and getAudit() methods.

Automated? Yes

Preconditions for Test: An IRVAudit object and IRV object are created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to make sure that the audit member is not nullptr after setting the audit.	The IRV and IRVAudit objects created.	Should not return nullptr.	Correctly returns a non null value.	None.

Postconditions for Test: The audit object is deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_setAudit Name of Tester: Jacob

Test Description: Checks to ensure that the setAudit() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setAudit() and getAudit() methods.

Automated? Yes
Preconditions for Test: An IRVAudit object and IRV object are created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to make sure that the audit member is not nullptr after setting the audit.	The IRV and IRVAudit objects created.	Should not return nullptr.	Correctly returns a non null value.	None.

Postconditions for Test: The audit object is deleted.
—---
—---
Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_getRemainCandidatesTest Name of Tester: Jacob
Test Description: Checks to ensure that the getRemainCandidates() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses getRemainCandidates and setRemainCandidates methods.
Automated? Yes
Preconditions for Test: An IRV object is created and it is set to have 10 remainCandidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tests that the remainCandidates is indeed 10.	The IRV object.	Should return 10.	Returns 10 correctly.	None.

2	Checks that remainCandidates is -10 after setting it.	The IRV object.	Should return -10.	Returns -10 correctly.	None.

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_setRemainCandidatesTest Name of Tester: Jacob

Test Description: Checks to ensure that the setRemainCandidates() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses getRemainCandidates() and setRemainCandidates() methods.

Automated? Yes

Preconditions for Test: An IRV object is created and it is set to have 10 remainCandidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tests that the remainCandidates is indeed 10.	The IRV object.	Should return 10.	Returns 10 correctly.	None.
2	Checks that remainCandidates is -10 after setting it.	The IRV object.	Should return -10.	Returns -10 correctly.	None.

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getNumCandidatesTest Name of Tester: Jacob
Test Description: Checks to ensure that the getNumCandidates() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses getNumCandidates() and setNumCandidates() methods.
Automated? Yes
Preconditions for Test: An IRV object is created and it is set to have 10 numCandidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tests that the numCandidates is indeed 10.	The IRV object.	Should return 10.	Returns 10 correctly.	None.
2	Checks that numCandidates is -10 after setting it.	The IRV object.	Should return -10.	Returns -10 correctly.	None.

Postconditions for Test: None.
—---
—---
Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_setNumCandidatesTest Name of Tester: Jacob
Test Description: Checks to ensure that the setNumCandidates() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses getNumCandidates() and setNumCandidates() methods.
Automated? Yes
Preconditions for Test: An IRV object is created and it is set to have 10 numCandidates.

Step #	Test Step	Test Data	Expected Result	Actual Result	Notes

	Description				
1	Tests that the numCandidates is indeed 10.	The IRV object.	Should return 10.	Returns 10 correctly.	None.
2	Checks that numCandidates is -10 after setting it.	The IRV object.	Should return -10.	Returns -10 correctly.	None.

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getBallotList Name of Tester: Jacob

Test Description: Checks to ensure that the getBallotList() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses appendNode(), setBallotList(), and getBallotList() methods.

Automated? Yes

Preconditions for Test: An IRV object and BallotLinkedList object is created. The list has one node and then is set to the field of the IRV object.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the list has a length of 1, using getBallotList.	The IRV and BallotLinkedList objects.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: Deletes the BallotLinkedList.

—---

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_setBallotList Name of Tester: Jacob
Test Description: Checks to ensure that the setBallotList() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses appendNode(), setBallotList(), and getBallotList() methods.
Automated? Yes
Preconditions for Test: An IRV object and BallotLinkedList object is created. The list has one node and then is set to the field of the IRV object.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the list has a length of 1, using setBallotList.	The IRV and BallotLinkedList objects.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: Deletes the BallotLinkedList.
—---
—---
Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_setCandidates Name of Tester: Jacob
Test Description: Checks to ensure that the setCandidates() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), setCandidates(), and getCandidates() methods.
Automated? Yes
Preconditions for Test: Two candidates objects are created and put into a vector. This is assigned to the field of an IRV object.

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

1	Checks that the index of the first candidate in the IRV list is 1.	The IRV object and candidate vector.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getCandidates Name of Tester: Jacob

Test Description: Checks to ensure that the getCandidates() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), setCandidates(), and getCandidates() methods.

Automated? Yes

Preconditions for Test: Two candidates objects are created and put into a vector. This is assigned to the field of an IRV object.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the index of the first candidate in the IRV list is 1.	The IRV object and candidate vector.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_printResults Name of Tester: Jacob

Test Description: Checks to ensure that the printResults() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setNumCandidates(), incInitialFirstVotes(0, setName(), incGainedVotes(), setCandidates(), and printResults() methods.
Automated? Yes
Preconditions for Test: Three candidates are created and given first place votes values and names. They are added to an IRV object. They are given specific gainedVotes values to ensure a correct output.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	printResults() is called, and the output is compared to the output string that is expected.	The IRV object created.	Expected string compare to return 0.	String compare returns -32.	This test fails, likely due to bugs in the code to print the information for the election.

Postconditions for Test: None.
—---
—---
Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_fillFirstPlaceVotes Name of Tester: Jacob
Test Description: Checks to ensure that the fillFirstPlaceVotes() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setCandidates(), populateList(), setBallotList(), and FillFirstPlaceVotes() methods.
Automated? Yes
Preconditions for Test: Three candidates are created and put in a vector. They are assigned to an IRV object. A BallotLinkedList vector is created, and it is filled with 4 BallotLinkedLists. This vector is also assigned to the IRV object. fillFirstPlaceVotes() is then called on the IRV object.

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

1	Checks that the first candidate has 0 initialFirstVotes.	The candidates vector in the IRV object.	Should return 0.	Returns 0 correctly.	None.
2	Checks that the first candidate has 0 workingVotes.	The candidates vector in the IRV object.	Should return 0.	Returns 0 correctly.	None.
3	Checks that the second candidate has 2 initialFirstVotes.	The candidates vector in the IRV object.	Should return 2.	Returns 2 correctly.	None.
4	Checks that the second candidate has 2 workingVotes.	The candidates vector in the IRV object.	Should return 2.	Returns 2 correctly.	None.
5	Checks that the third candidate has 1 workingVotes.	The candidates vector in the IRV object.	Should return 1.	Returns 1 correctly.	None.
6	Checks that the third candidate has 1 workingVotes.	The candidates vector in the IRV object.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: The list of BallotLinkedLists is deleted.
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: ElectionEntity_setName Name of Tester: Ruolei
Test Description: Checks to ensure that the setName(), getName() method in ElectionEntity.cc works correctly.

Test Storage Location and Functions Used: Found in ElectionEntity_unittest.cc and uses setName, getNamemethods in ElectionEntity
Automated? Yes
Preconditions for Test: A ElectionEntity is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the electionEntity set the correct Name which should be test123	Operates on electionEntity,set the name “test123”.	Should return test123.	Returns test123 correctly.	None.

Postconditions for Test: The ElectionEntity objects that were created are deleted.
—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Party_ incrementVotes Name of Tester: Ruolei
Test Description: Checks to ensure that the incrementVotes (), getNumVotes () method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses incrementVotes, getNumVotes in Party
Automated? Yes
Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct numVotes 0	Operates on party.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Party have the correct numVotes 1 after calling incrementVotes	Operates on party and call incrementVotes	Should return 1.	Returns 1 correctly.	None.
3	Checks to see if the Party have the correct numVotes 2 After calling incrementVotes	Operates on party and call incrementVotes	Should return 2.	Returns 2 correctly.	None.

Postconditions for Test: The Party objects that were created are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Party_ setFirstSeats Name of Tester: Ruolei
Test Description: Checks to ensure that the setFirstSeats(), getFirstSeats () method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses setFirstSeats, getFirstSeats in Party
Automated? Yes
Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct FirstSeats 0	Operates on party.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Party have the correct FirstSeats 100 after calling setFirstSeats 100	Operates on party and call setFirstSeats	Should return 100.	Returns 100 correctly.	None.
3	Checks to see if the Party have the correct FirstSeats After calling setFirstSeats -100	Operates on party and call setFirstSeats	Should return -100.	Returns -100 correctly.	None.

Postconditions for Test: The Party objects that were created are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Party_ setSecondSeats Name of Tester: Ruolei
Test Description: Checks to ensure that the setSecondSeats (), getSecondSeats () method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses setSecondSeats, getSecondSeats in Party
Automated? Yes

Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct SecondSeats 0	Operates on party.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Party have the correct SecondSeats100 after calling setSecondSeats 100	Operates on party and call setSecondSeats	Should return 100.	Returns 100 correctly.	None.
3	Checks to see if the Party have the correct SecondSeats After calling setSecondSeats -100	Operates on party and call setSecondSeats	Should return -100.	Returns -100 correctly.	None.

Postconditions for Test: The Party objects that were created are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Party_ setNumVotes Name of Tester: Ruolei

Test Description: Checks to ensure that the setNumVotes (), getNumVotes () method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses setNumVotes, getNumVotes in Party
Automated? Yes
Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct numVotes 0	Operates on party.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Party have the correct numVotes 2 after calling setNumVotes 2	Operates on party and call setNumVotes	Should return 2.	Returns 2 correctly.	None.
3	Checks to see if the Party have the correct numVotes After calling setNumVotes -2	Operates on party and call setNumVotes	Should return -2.	Returns -2 correctly.	None.

Postconditions for Test: The Party objects that were created are deleted.
—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Party_ setLeftoverVotes Name of Tester: Ruolei

Test Description: Checks to ensure that the setLeftoverVotes (), getLeftoverVotes () method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses setNumVotes, getLeftoverVotes in Party
Automated? Yes
Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct leftoverVotes 0	Operates on party.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Party have the correct leftoverVotes 2 after calling setLeftoverVotes 2	Operates on party and call setLeftoverVotes	Should return 2.	Returns 2correctly.	None.
3	Checks to see if the Party have the correct leftoverVotes After calling setLeftoverVotes -2	Operates on party and call setLeftoverVotes	Should return -2.	Returns -2 correctly.	None.

Postconditions for Test: The Party objects that were created are deleted.
—--
—--

Test Stage: Unit Test Date: 3/25/23

Test Case ID: Party_ setCandidates Name of Tester: Ruolei
Test Description: Checks to ensure that the setCandidates (), getCandidates () getNumCandidates() method in Party.cc works correctly.
Test Storage Location and Functions Used: Found in Party_unittest.cc and uses setCandidates, getCandidates, getNumCandidates in Party
Automated? Yes
Preconditions for Test: A Party is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Party have the correct candidates	Operates on party.	Should return “”.	Returns “” correctly.	None.
2	Checks to see if the Party have the correct candidates after calling setCandidates “test1”	Operates on party and call setLeftoverVotes	Should return “test1”.	Returns “test1” correctly.	None.
3	Checks to see if the Party have the correct candidates after calling setCandidates “test1,test2”	Operates on party and call setCandidates	Should return “test1, test2”.	Returns “test1, test2”correctly.	None.
4	Checks to see if the Party have the correct candidates number after calling setCandidates “test1,test2”	Operates on party and call setCandidates	Should return 2	Returns 2 correctly.	None

Postconditions for Test: The Party objects that were created are deleted.

—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_IncWorkingVotes Name of Tester: Ruolei
Test Description: Checks to ensure that the IncWorkingVotes (), getWorkingVotes ()method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses IncWorkingVotes, getWorkingVotes, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Candidate have the correct workingVotes 0	Operates on candidate.	Should return 0.	Returns 0 correctly.	None.

2	Checks to see if the Candidate have the correct workingVotes 1 after calling incWorkingVotes	Operates on candidate and call incWorkingVotes 1	Should return 1.	Returns 1 correctly.	None.
3	Checks to see if the Candidate have the correct workingVotes 10001 After calling incWorkingVotes 10000	Operates on candidate and call incWorkingVotes 10000	Should return 10001.	Returns 10001 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_ SetWorkingVotes   Name of Tester: Ruolei
Test Description: Checks to ensure that the SetWorkingVotes (), getWorkingVotes ()method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses SetWorkingVotes, getWorkingVotes, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

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

1	Checks to see if the Candidate have the correct workingVotes 0	Operates on candidate.	Should return 0.	Returns 0 correctly.	None.
2	Checks to see if the Candidate have the correct workingVotes 10 after calling SetWorkingVotes	Operates on candidate and call SetWorkingVotes 10	Should return 10.	Returns 10 correctly.	None.
3	Checks to see if the Candidate have the correct SetWorkingVotes -10 After calling SetWorkingVotes -10	Operates on candidate and call SetWorkingVotes -10	Should return -10.	Returns -10 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_ SetIndex Name of Tester: Ruolei
Test Description: Checks to ensure that the setIndex (), getIndex ()method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses setIndex, getIndex, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Candidate have the correct index -1	Operates on candidate.	Should return -1.	Returns -1 correctly.	None.
2	Checks to see if the Candidate have the correct index 10 after calling setIndex	Operates on candidate and call setIndex 10	Should return 10.	Returns 10 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.

—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_IncInitialFirstVotes Name of Tester: Ruolei
Test Description: Checks to ensure that the IncInitialFirstVotes (), getInitialFirstVotes ()method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses IncInitialFirstVotes, getInitialFirstVotes, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Candidate have the	Operates on candidate.	Should return 0.	Returns 0 correctly.	None.

	correct initialFirstVotes 0				
2	Checks to see if the Candidate have the correct initialFirstVotes 10 after calling IncInitialFirstVotes	Operates on candidate and call IncInitialFirstVotes 10	Should return 10.	Returns 10 correctly.	None.
3	Checks to see if the Candidate have the correct initialFirstVotes 8 after calling IncInitialFirstVotes	Operates on candidate and call IncInitialFirstVotes -2	Should return 8.	Returns 8 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_ IncGainedVotes Name of Tester: Ruolei
Test Description: Checks to ensure that the IncGainedVotes(), getGainedVotes ()method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses IncGainedVotes, IncGainedVotes, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the Candidate array have the correct gainedVotes 0	Operates on candidate.	Should return 0.	Returns 0 correctly.	None.
3	Checks to see if the Candidate array have the correct gainedVotes 1 after calling IncGainedVotes	Operates on candidate and call IncGainedVotes every array element index	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.
—---
—---

Test Stage: Unit Test Date: 3/25/23
Test Case ID: Candidate_ setGainedVotes Name of Tester: Ruolei
Test Description: Checks to ensure that the setGainedVotes() method in Candidate.cc works correctly.
Test Storage Location and Functions Used: Found in Candidate_unittest.cc and uses IncGainedVotes, IncGainedVotes, in Candidate
Automated? Yes
Preconditions for Test: A Candidate is created

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

1	Checks to make sure an error doesnt occur when index of -1 is used.	Operates on candidate.	Should run with no errors.	Runs with no errors correctly.	None.
2	Checks to make sure no error occurs when an out of bounds index is used.	Operates on candidate.	Should run with no errors.	Runs with no errors correctly.	None.
3	Checks that the index of gained votes is -1 when the method is called.	Operates on candidate.	Should return -1.	Returns -1 correctly.	None.

Postconditions for Test: The Candidate objects that were created are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getWinnerTwoLeft Name of Tester: Jacob

Test Description: Checks to ensure that the getWinnerTwoLeft() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), incWorkingVotes(), setCandidates() methods.

Automated? Yes

Preconditions for Test: An IRV object has its fields initialized with three candidate objects. One is the clear winner. One is out of the running.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to ensure the winner is not nullptr after calling getWinnerTwoLeft( ).	The IRV object initialized before.	Should return anything other than nullptr.	Returns a non nullptr value correctly.	None.
2	Checks that the working votes of the winning candidate is 1000.	The IRV object initialized before.	Should return 1000.	Returns 1000 correctly.	None.
3	Checks that the winner is not nullptr when a tie needs to be broken.	A new IRV object where the two candidates are tied.	Should return nullptr.	Test seg faults.	This is likely due to an issue in the writeTieLead() function.
4	Checks that the index of the winning candidate is 69.	The new IRV object.	Should return 69.	Test seg faults.	This is likely due to an issue in the writeTieLead() function.

Postconditions for Test: Audit and stream objects are deleted.

—--

—--

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_maxCandWinCheck Name of Tester: Jacob

Test Description: Checks to ensure that the maxCandWinCheck() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), incWorkingVotes(), setCandidates() methods.

Automated? Yes

Preconditions for Test: An IRV object has its fields initialized with three candidate objects. The candidate with the most votes should win.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to ensure the winner is not nullptr after call to maxCandWinCheck	The IRV object initially created.	Should return non nullptr.	Correctly returns non nullptr.	None.
2	Checks that the working votes of the winner is 1000.	The IRV object initially created.	Should return 1000.	Returns 1000 correctly.	None.
3	Checks who the winner is after excluding obvious winner.	The IRV object initially created.	Should return 500.	Returns 500 correctly.	None.
4	Checks that the winner is not nullptr.	The IRV object initially created.	Should return non nullptr.	Returns non nullptr correctly.	None.
5	Checks that the winner is nullptr when no candidate is clear winner.	New IRV object with no candidate having majority.	Should return nullptr.	Returns nullptr correctly.	None.

Postconditions for Test: None.

—--

—--

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_getLoser Name of Tester: Jacob
Test Description: Checks to ensure that the getLoser() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), incWorkingVotes(), setCandidates(), getLoser() methods.
Automated? Yes
Preconditions for Test: An IRV object has its fields initialized with three candidate objects. One candidate is the clear loser.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the calculated loser has 999 workingVotes.	Initial IRV object.	Should return 999.	Returns 999 correctly.	None.
2	Checks that the loser has 1000 working votes in a new IRV object.	Second IRV object.	Should return 1000.	Returns 1000 correctly.	None.
3	Checks that the loser has 5000 workingVotes in a tie situation.	Third IRV object.	Should return 5000.	Returns 5000 correctly.	None.

Postconditions for Test: None.

—--

—--

Test Stage: Unit Test Date: 3/25/23
Test Case ID: IRV_redistribute Name of Tester: Jacob
Test Description: Checks to ensure that the redistribute() method in IRV works correctly.
Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), incWorkingVotes(), setCandidates(), redistribute() methods.

Automated? Yes
Preconditions for Test: An IRV object has its fields initialized with four candidate objects. One candidate is the clear loser with two votes.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks that the head of the first ballot list is 0.	IRV object initially created and filled.	Should return 0.	Returns 0 correctly.	None.
2	Checks that the head of the second ballot list is 3.	IRV object initially created and filled.	Should return 3.	Returns 3 correctly.	None.
3	Checks that the head of the third ballot list is 2.	IRV object initially created and filled.	Should return 2.	Returns 2 correctly.	None.
4	Checks that the head of the fourth ballot list is -1.	IRV object initially created and filled.	Should return -1.	Returns -1 correctly.	None.
5	Checks that the working votes of the first candidate is 11.	IRV object initially created and filled.	Should return 11.	Returns 11 correctly.	None.
6	Checks that the working votes of the second candidate is 2.	IRV object initially created and filled.	Should return 2.	Returns 2 correctly.	None.

7	Checks that the working votes of the third candidate is 12.	IRV object initially created and filled.	Should return 12.	Returns 12 correctly.	None.
8	Checks that the working votes of the fourth candidate is 14.	IRV object initially created and filled.	Should return 14.	Returns 14 correctly.	None.
9	Checks that the gained votes of the first candidate is 1.	IRV object initially created and filled.	Should return 1.	Returns 1 correctly.	None.
10	Checks that the gained votes of the second candidate is 1.	IRV object initially created and filled.	Should return 0.	Returns 0 correctly.	None.
11	Checks that the gained votes of the third candidate is 1.	IRV object initially created and filled.	Should return 0.	Returns 0 correctly.	None.
12	Checks that the gained votes of the fourth candidate is 1.	IRV object initially created and filled.	Should return 1.	Returns 1 correctly.	None.

Postconditions for Test: The linked list vector and audit are deleted.

—---

—---

Test Stage: Unit Test Date: 3/25/23

Test Case ID: IRV_calculateWinners Name of Tester: Jacob

Test Description: Checks to ensure that the calculateWinners() method in IRV works correctly.

Test Storage Location and Functions Used: Located in the IRV_unittest.cc file and uses setIndex(), incWorkingVotes(), setCandidates(), calculateWinners() methods.
Automated? Yes
Preconditions for Test: An IRV object has its fields initialized with four candidate objects. Four ballot lists are created and the first candidate should win.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to ensure the call to calculateWinners completes with no errors.	The IRV object initialized.	Should run with no errors.	Runs with no errors correctly.	None.
2	Checks that the first candidate has three workingVotes.	The IRV object initialized.	Should return 3.	Returns 3 correctly.	None.
3	Checks that the first candidate has three initialFirstVotes.	The IRV object initialized.	Should return 3.	Returns 3 correctly.	None.
4	Checks that the workingVotes of the first candidate is 4.	New IRV object where redistribution takes place.	Should return 4.	Returns 4 correctly.	None.
5	Checks that the initialFirstVotes of the first candidate is 3.	New IRV object where redistribution takes place.	Should return 3.	Returns 3 correctly.	None.
6	Checks that the	New IRV object	Should return 1.	Returns 1 correctly.	None.

	gained votes of the first candidate is 1.	where redistribution takes place.			
7	Checks that the workingVotes of the second candidate is 2.	New IRV object where redistribution takes place.	Should return 2.	Returns 2 correctly.	None.
8	Checks that the initialFirstVotes of the second candidate is 2.	New IRV object where redistribution takes place.	Should return 2.	Returns 2 correctly.	None.
9	Checks that the gained votes of the second candidate is 0.	New IRV object where redistribution takes place.	Should return 0.	Returns 0 correctly.	None.
10	Checks that the workingVotes of the third candidate is 1.	New IRV object where redistribution takes place.	Should return 1.	Returns 1 correctly.	None.
11	Checks that the initialFirstVotes of the third candidate is 3.	New IRV object where redistribution takes place.	Should return 3.	Returns 3 correctly.	None.
12	Checks that the gained votes of the third candidate is 0.	New IRV object where redistribution takes place.	Should return 0.	Returns 0 correctly.	None.

Postconditions for Test: The linked list vectors and audist are deleted.

—---

—---

Test Stage: System Test Date: 3/26/23
Test Case ID: CPL_system1 Name of Tester: Jacob
Test Description: Runs the CPL algorithm on the file CPLTest1.csv, which is a simple CPL example with no ties and two allocations.
Test Storage Location and Functions Used: Test file is located in the testing directory. The program itself was ran.
Automated? No
Preconditions for Test: CPLTest1.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Output to the terminal is verified to be correct based on the CPL algorithm.	CPLTest1.csv	Democratic party should get 1 seat initially with the winner being Foster. On the second allocation Republican and New wave should get a seat, which their winners being Green and Mcclure.	Output matches what was expected.	None.
2	Audit file contents were verified to ensure they were what was expected.	CPLTest1.csv	We expect the audit file to trace all steps of the election in a correct and well formatted manner, with results matching the terminal.		None

Postconditions for Test: CPLAudit_12_12_2001.txt is located in the audits folder.

—---
—---

Test Stage: System Test Date: 3/26/23
Test Case ID: IRV_system1 Name of Tester: Jacob
Test Description: Runs the IRV algorithm on the file IRVTest1.csv, which is a simple IRV example with no ties and two redistributions.
Test Storage Location and Functions Used: Test file is located in the testing directory. The program itself was ran.
Automated? No
Preconditions for Test: IRVTest1.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The output which was given on the screen was manually verified.	IRVTest1.csv	Rosen should be the winner, and there should be two redistributions with Rosen gaining one vote.	Output is as expected.	None
2	The audit file output was verified.	IRVTest1.csv	We expect all steps of the audit process to be present and in a readable format.	All output is present and readable.	None.

Postconditions for Test: IRVAudit_12_12_2001.txt is located in the audits folder.

—---
—---

Test Stage: System Test Date: 3/26/23
Test Case ID: IRV_system2 Name of Tester: Jacob
Test Description: Runs the IRV algorithm on the file IRVTest2.csv, which is a simple IRV example with one tie to see who has votes redistributed. Ran several times to ensure equal chance of winning the tie.

Test Storage Location and Functions Used: Test file is located in the testing directory. The program itself was ran.
Automated? No
Preconditions for Test: IRVTest2.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Ran the program several times to ensure that each candidate lost the tie around half the time.	IRVTest2.csv	Kleinberg and Royce should each be eliminated in some of the runs of the program.	Kleinberg and Royce were both eliminated some of the time.	None.
2	Verify audit file contents.	IRVTest2.csv	The audit should report all info accurately.	All info is present and accurate, including report of tie breaker.	None.

Postconditions for Test: IRVAudit_12_12_2001.txt is located in the audits folder.
—---
—---
Test Stage: System Test Date: 3/26/23
Test Case ID: IRV_system3 Name of Tester: Jacob
Test Description: Runs the IRV algorithm on the file IRVTest3.csv, which is a IRV example with a three way tie for the loser. Ran several times to ensure the loser could be any of the three.
Test Storage Location and Functions Used: Test file is located in the testing directory. The program itself was ran.
Automated? No
Preconditions for Test: IRVTest3.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step	Test Data	Expected Result	Actual Result	Notes

	Description				
1	Tested to ensure each of the three was the loser some of the time.	IRVTest3.csv	Each of the three (McIsaac, Kleinberg, and Royce) should lose some of the time.	They each lose some of the time.	None
2	Verified audit output.	IRVTest3.csv	Ensures all audit info is present and accurate.	All info is present and accurate.	None

Postconditions for Test: IRVAudit_12_12_2001.txt is located in the audits folder.

—---

—---

Test Stage: System Test Date: 3/26/23

Test Case ID: IRV_system4 Name of Tester: Jacob

Test Description: Runs the IRV algorithm on the file IRVTest4.csv, which is a IRV example with a two way tie for the winner.

Test Storage Location and Functions Used: Test file is located in the testing directory. The program itself was ran.

Automated? No

Preconditions for Test: IRVTest4.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tested to ensure each of the two candidates wins some of the time.	IRVTest4.csv	Each of the two (Rosen and Kleinberg) should win some of the time.	They each win some of the time.	None
2	Verified audit	IRVTest4.csv	Ensures all audit	All info is present	None

	output.		info is present and accurate.	and accurate.	

Postconditions for Test: IRVAudit_12_12_2001.txt is located in the audits folder.

—---

—---

Test Stage: System Test Date: 3/26/23
Test Case ID: user_input Name of Tester: Jacob
Test Description: Tests to make sure that the program can accept file names on the command line and by prompting. Checks to make sure that erroneous input is handled gracefully.
Automated? No
Preconditions for Test: votingMain.out is present.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to make sure the program works when file name given on command line and in a prompt.	Any existing csv file.	Program should run to completion in both cases.	Runs to completion in both.	None.
2	Check to see if program handles erroneous input.	Any existing csv file.	Program should reprompt user when given bad input.	Erroneous input is handled except in the case where the file does not exist (the program ends), the year is in the future, and when the day and month are incompatible.	

Postconditions for Test: IRVAudit_12_12_2001.txt is located in the audits folder.
—--
—--
Test Stage: Unit Test Date: 3/26/23
Test Case ID: ElectionAudit_setAuditFile Name of Tester: Caleb
Test Description: Tests to ensure the setAuditFile() function is working properly.
Test Storage Location and Functions Used: Found in ElectionAudit_unittest.cc and uses setAuditFile() and getAuditFile() which are methods of the ElectionAudit class.
Automated? Yes
Preconditions for Test: An election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None

Postconditions for Test: Filestream redirected properly and all objects deleted.
—--
—--
Test Stage: Unit Test Date: 3/26/23
Test Case ID: IRVAudit_writeMetadata Name of Tester: Caleb
Test Description: Tests to ensure the writeMetadata() function is working properly.
Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), IRV(), setNumCandidates(), setNumBallots(), Candidate()(Candidate and IRV Classes), writeMetaData(), and IRVAudit()(IRVAudit Class).
Automated? Yes

Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of candidates and ballots so that there is metaData to be written.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object instantiated	Uses the IRV Class	Creates an IRV Object	Creates an IRV Object	None
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare metaData	Defines the candidates, numCandidates, and numBallots for skeleton IRV.	Declares all data correctly so that it outputs when called in writeMetadata functions	Declares all data correctly so that it outputs when called in writeMetadata functions	None
5	Call writeMetaData and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: IRVAudit_writeFinalResults Name of Tester: Caleb
Test Description: Tests to ensure the writeFinalResults() function is working properly.
Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), IRV(), setNumCandidates(), setNumBallots(), Candidate(), setName()(Candidate and IRV Classes), writeFinalResults(), and IRVAudit()(IRVAudit Class).
Automated? Yes
Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of named candidates and one winning candidate so that results can be written.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object instantiated	Uses the IRV Class	Creates an IRV Object	Creates an IRV Object	None
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare winning candidate.	Defines the winning candidate and sets their name	Winning candidate is named “Bob”	Winning Candidate is named “Bob”	None
5	Call writeFinalResults and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.
—---
—---
Test Stage: Unit Test Date: 3/26/23
Test Case ID: IRVAudit_writeBallot Name of Tester: Caleb
Test Description: Tests to ensure the writeBallots() function is working properly.
Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), IRV(), setNumCandidates(), setNumBallots(), Candidate(), setName()(Candidate and IRV Classes), writeBallots(), and IRVAudit()(IRVAudit Class).
Automated? Yes
Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of named candidates and a vector of votes defining the order of choices on the ballot.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object instantiated	Uses the IRV Class	Creates an IRV Object	Creates an IRV Object	None
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare named candidates vector	Defines the candidate vectors	Candidate vector has (Bob, Steve,	Candidate vector has (Bob, Steve,	None

	and vote vector	and votes vector with proper data	Tres) and votes has (1, 2, 3)	Tres) and votes has (1, 2, 3)	
5	Call writeBallot and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: IRVAudit_writeVoteTransfer Name of Tester: Caleb

Test Description: Tests to ensure the writeVoteTransfer() function is working properly.

Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), IRV(), setNumCandidates(), setNumBallots(), Candidate(), setName(), incGainedVotes()(Candidate and IRV Classes), writeBallots(), and IRVAudit()(IRVAudit Class).

Automated? Yes

Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of named candidates, one eliminated candidate, a vector of who is running, and the votes redistributed from the elimination are logged.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object	Uses the IRV Class	Creates an IRV	Creates an IRV	None

	instantiated		Object	Object	
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare named candidates vector and running vector	Defines the candidate vectors and running vector with proper data	Candidate vector has (Bob, Steve, Tres) and running has (true, false, true)	Candidate vector has (Bob, Steve, Tres) and running has (true, false, true)	None
5	Declare eliminated candidate and log gained votes	Set Bob as eliminated and increase gained votes of others	Bob is declared eliminated and gained votes of other candidates increased by 1.	Bob is declared eliminated and gained votes of other candidates increased by 1.	None
6	Call writeVoteTransfer and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: IRVAudit_writeTieLead Name of Tester: Caleb
Test Description: Tests to ensure the writeTieLead() function is working properly.
Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), ElectionEntity()(Election Entity class), IRV(), setNumCandidates(), Candidate(), setName()(Candidate and IRV Classes), writeTieLead(), IRVAudit()(IRVAudit Class).
Automated? Yes

Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of named candidates of type ElectionEntity, a vector of those in the tie is created as well as who won the tie.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object instantiated	Uses the IRV Class	Creates an IRV Object	Creates an IRV Object	None
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare named ElectionEntity candidates vector	Defines the candidate vector with proper data	Candidate vector has (Bob, Steve, Tres)	Candidate vector has (Bob, Steve, Tres)	None
5	Declare ElectionEntity tieArray vector and tieWinner vector	Set Steve as winner and put steve and Bob in tieArray vector	Steve is declared winner in tie breaker against Bob	Steve is declared winner in tie breaker against Bob	None
6	Call writeTieLead and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.
—---

—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: IRVAudit_writeTieLose Name of Tester: Caleb
Test Description: Tests to ensure the writeTieLose() function is working properly.
Test Storage Location and Functions Used: Found in IRVAudit_unittest.cc and uses setAuditFile()(Election Audit Class), ElectionEntity()(Election Entity class), IRV(), setNumCandidates(), Candidate(), setName()(Candidate and IRV Classes), writeTieLose(), IRVAudit()(IRVAudit Class).
Automated? Yes
Preconditions for Test: An IRV election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. An IRV election object is created with a number of named candidates of type ElectionEntity, a vector of those in the tie is created as well as who won the tie.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	IRV election object instantiated	Uses the IRV Class	Creates an IRV Object	Creates an IRV Object	None
3	IRV audit file assigned	Declares IRVAudit file for writing to	Creates IRVAudit object	Creates IRVAudit object	None
4	Declare named ElectionEntity candidates vector	Defines the candidate vector with proper data	Candidate vector has (Bob, Steve, Tres)	Candidate vector has (Bob, Steve, Tres)	None
5	Declare ElectionEntity	Set Bob as winner and put steve and	Bob is declared winner in tie	Bob is declared winner in tie	None

	tieArray vector and tieWinner vector	Bob in tieArray vector	breaker for lowest votes against Steve	breaker for lowest votes against Steve	
6	Call writeTieLose and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeMetadata Name of Tester: Caleb

Test Description: Tests to ensure the writeMetadata() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots()(Party and CPL Classes), writeMetadata(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object	Uses the CPL Class	Creates a CPL	Creates a CPL	None

	instantiated		Object	Object	
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each	Creates two parties(Republican and Democratic) with two respective candidates	Creates two parties(Republican and Democratic) with two respective candidates	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None
6	Call writeMetadata and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeInitialVotes Name of Tester: Caleb

Test Description: Tests to ensure the writeInitialVotes() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes()(Party and CPL Classes), writeInitalVotes(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None
6	Call writeInitalVotes and compare with	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

	expected string				

Postconditions for Test: Filestream redirected properly and all objects deleted.

—--

—--

Test Stage: Unit　　　　　　　　　　　　　　　　　　　　　　　　Test Date: 3/26/23
Test Case ID: CPLAudit_writeFinalResults　　　　　　　　　　　　　　Name of Tester: Caleb
Test Description: Tests to ensure the writeFinalResults() function is working properly.
Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats()(Party and CPL Classes), writeFinalResults(), CPLAudit()(CPLAudit Class).
Automated? Yes
Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None

4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave two seats to Democratic(1 each round) and none to Republican.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave two seats to Democratic(1 each round) and none to Republican.	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None
6	Call writeFinalResults and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeFirstAllocation Name of Tester: Caleb

Test Description: Tests to ensure the writeFirstAllocation() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats()(Party and CPL Classes), writeFirstAllocation(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave 1 seat to Democratic and none to Republican.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave 1 seat to Democratic and none to Republican.	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None

6	Call writeFirstAllocation and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeSecondAllocation Name of Tester: Caleb

Test Description: Tests to ensure the writeSecondAllocation() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats(), setLeftoverVotes()(Party and CPL Classes), writeSecondAllocation(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None

3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave 1 seat to Democratic in second allocate with 1 remainder vote. Gave 0 seat to Republican but 1 remainder vote	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party. Gave 1 seat to Democratic in second allocate with 1 remainder vote. Gave 0 seat to Republican but 1 remainder vote	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None
6	Call writeSecondAllocation and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeLottery Name of Tester: Caleb

Test Description: Tests to ensure the writeLottery() function is working properly.
Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats(), setLeftoverVotes()(Party and CPL Classes), writeLottery(), CPLAudit()(CPLAudit Class).
Automated? Yes
Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	None
5	Declare parties to election and	Add in parties, and numSeats and	Parties are added to election, 2 seats are	Parties are added to election, 2 seats are	None

	election/ballot details	Ballots.	allocated, and 2 Ballots are recorded	allocated, and 2 Ballots are recorded	
6	Call writeLottery and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string with seats being lotteried from Democratic to Republican	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeBallot Name of Tester: Caleb

Test Description: Tests to ensure the writeBallot() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats(), setLeftoverVotes()(Party and CPL Classes), writeBallot(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester	Returns string correctly	None

			string		
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots.	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded	None
6	Call writeBallot and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string with Ballot being written for Democratic	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeTiebreaker Name of Tester: Caleb

Test Description: Tests to ensure the writeTiebreaker() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats(), setLeftoverVotes()(Party and CPL Classes), writeTieBreaker(), CPLAudit()(CPLAudit Class), and ElectionEntity() (Election Entity class).
Automated? Yes
Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	None
5	Declare parties to election and	Add in parties, and numSeats and	Parties are added to election, 2 seats are	Parties are added to election, 2 seats are	None

	election/ballot details	Ballots.	allocated, and 2 Ballots are recorded	allocated, and 2 Ballots are recorded	
6	Declare ElectionEntity tieArray vector and tieWinner vector	Set Bob as winner and put steve and Bob in tieArray vector	Bob is declared winner in tie breaker for lowest votes against Steve	Bob is declared winner in tie breaker for lowest votes against Steve	None
7	Call writeTiebreaker and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string with Bob winning tiebreaker	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: CPLAudit_writeQuota Name of Tester: Caleb

Test Description: Tests to ensure the writeQuota() function is working properly.

Test Storage Location and Functions Used: Found in CPLAudit_unittest.cc and uses setAuditFile()(Election Audit Class), CPL(), setNumParties(), Party(), setName(), setCandidates(), setParties(), setSeats(), setNumBallots(), setNumVotes(), setFirstSeats(), setSecondSeats(), setLeftoverVotes()(Party and CPL Classes), writeQuota(), CPLAudit()(CPLAudit Class).

Automated? Yes

Preconditions for Test: A CPL election audit file stream is created and is redirected to stdout so that it can be captured and compared to ensure that data is outputting properly. A CPL election object is created with a number of named parties with candidates. The number of seats and earned seats, parties, and ballots are all then set to the election.

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

1	Redirect filestream to stdout	Uses these file streams	The data written to filestream should be equal to the tester string	Returns string correctly	None
2	CPL election object instantiated	Uses the CPL Class	Creates a CPL Object	Creates a CPL Object	None
3	CPL audit file assigned	Declares CPLAudit file for writing to	Creates CPLAudit object	Creates CPLAudit object	None
4	Declare parties and details	Defines the parties vector with two candidates each and number of votes and First/Second seats	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	Creates two parties(Republican and Democratic) with two respective candidates. With 2 votes to each party.	None
5	Declare parties to election and election/ballot details	Add in parties, and numSeats and Ballots and quota	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded. Quota is 1	Parties are added to election, 2 seats are allocated, and 2 Ballots are recorded. Quota is 1	None
6	Call writeQuota and compare with expected string	Writes to filestream correctly and captured by stdout	Written data is same as expected string	Written data is same as expected string with quota being 1	None

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: System   Test Date: 3/26/23

Test Case ID: CPL_system2 Name of Tester: Jacob
Test Description: Runs the CPL algorithm on the file CPLTest3.csv, which is a simple CPL example with a tie on the last seat to be allocated.
Automated? No
Preconditions for Test: CPLTest3.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The output should show that two of the three parties earned a seat that were tied.	CPLTest3.csv	Two of the three parties that were tied should have been given a seat.	Output matches what was expected.	None.
2	Verfies that audit contents are correct.	CPLTest3.csv	The audit should report the tie and the parties involved.	Output matches what was expected.	None.

Postconditions for Test: Filestream redirected properly and all objects deleted.
—---
—---
Test Stage: System Test Date: 3/26/23
Test Case ID: CPL_system2 Name of Tester: Jacob
Test Description: Runs the CPL algorithm on the file CPLTest3.csv, which is a simple CPL example with a tie on the last seat to be allocated.
Automated? No
Preconditions for Test: CPLTest3.csv is present in the same directory as votingMain.out, which has been compiled.

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

1	The output should show that two of the three parties earned a seat that were tied.	CPLTest3.csv	Two of the three parties that were tied should have been given a seat.	Output matches what was expected.	None.
2	Verfies that audit contents are correct.	CPLTest3.csv	The audit should report the tie and the parties involved.	Output matches what was expected.	None.

Postconditions for Test: Filestream redirected properly and all objects deleted.

—---

—---

Test Stage: System Test Date: 3/26/23
Test Case ID: CPL_system3 Name of Tester: Jacob
Test Description: Runs the CPL algorithm on the file CPLTest4.csv, which is a simple CPL example where two seats need to be lotteried off.
Automated? No
Preconditions for Test: CPLTest4.csv is present in the same directory as votingMain.out, which has been compiled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to make sure output shows two seats being lotteried to random parties.	CPLTest4.csv	The output should show that the Democrats earned 3 seats and two other parties should have a seat.	Output shows that the main party only has 3 out of the 5 earned seats, but the output does not show that other parties gained the seats.	None.

2	Verifies that audit contents are correct.	CPLTest4.csv	The audit should report the lottery of the seats.	The audit shows that seats are lotteried to parties, but that is not reflected in the final total for those parties.	None.

—---

—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: CPL_testSettersGetters Name of Tester: Daniel
Test Description: Verifies functionality of setters and getters of CPL class
Test Storage Location and Functions Used: Found in CPL_unittest.cc and uses setNumParties(), setSeats(), setAudit(), setParties(), getNumParties(), getSeats(), getAudit(), and getParties(), methods in CPL.
Automated? Yes
Preconditions for Test: Initialize a CPL object, Party object, and a CPLAudit pointer to a CPLAudit object.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Sets values of numParties, seats, parties, and audit using setters	CPL object, Party vector pointer to a Party vector, CPLAudit pointer to a CPLAudit object	Should set numParties to 5, seats to 10, parties to the Party vector pointer, and audit to the CPLAudit pointer.	Does the expected result.	No results expected for setting, both setters and getters are verified in step 2.
2	Checks that values set in step 1 are retrieved by getters	CPL object, Party object, CPLAudit pointer	Should return 5 for numParties, 10 for seats, Party vector	Does the expected result.	

		to a CPLAudit object	pointer for parties and CPLAudit pointer for the audit.		

Postconditions for Test: Deletes the Party vector (the CPLAudit object is handled by CPL's destructor).

—---

—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: CPL_testFirstAllocation Name of Tester: Daniel
Test Description: Verifies functionality of firstAllocation() of CPL class
Test Storage Location and Functions Used: Found in CPL_unittest.cc and uses firstAllocation and getParties(), methods in CPL, uses getFirstSeats() and getLeftoverVotes() in Party.
Automated? Yes
Preconditions for Test: Initialize a CPL object and give the object data equivalent to calling a handleCPL() in FileParser, as well as a precalculated quota, and create a boolean vector pointer that points to a boolean vector.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls firstAllocation	CPL object, int quota, boolean vector pointer to a boolean vector	Should set the parties firstSeats and leftoverVotes.	Does the expected result.	
2	Checks that getFirstSeats() of the parties returns the correct values	CPL object	Should return 3, 2, 2, 1, 0 for each party, respectively.	Does the expected result.	

3	Checks that getLeftoverVotes() of the parties returns the correct values	CPL object	Should return 8000, 3000, 1000, 2000, 6000, for each party, respectively.	Does the expected result.	

Postconditions for Test: Deletes the boolean vector.

—---
—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: CPL_testSecondAllocation Name of Tester: Daniel
Test Description: Verifies functionality of secondAllocation() of CPL class
Test Storage Location and Functions Used: Found in CPL_unittest.cc and uses secondAllocation and getParties(), methods in CPL.
Automated? Yes
Preconditions for Test: Initialize a CPL object and give the object data equivalent to calling a handleCPL() in FileParser, and create a boolean vector pointer that points to a boolean vector.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls firstAllocation	CPL object, int quota, boolean vector pointer to a boolean vector	Should set the parties firstSeats and leftoverVotes.	Does the expected result.	
2	Calls secondAllocation	CPL object, int quota, boolean vector pointer to a boolean	Should set the parties secondSeats.	Does the expected result.	

		vector			
3	Checks that getSecondSeats() of the parties returns the correct values	CPL object	Should return 1, 0, 0, 0, 1, for each party, respectively.	Does the expected result.	

Postconditions for Test: Deletes the boolean vector.

—---

—---

Test Stage: Unit   Test Date: 3/26/23

Test Case ID: CPL_testPrintResults()   Name of Tester: Daniel

Test Description: Verifies functionality of printResults() of CPL class

Test Storage Location and Functions Used: Found in CPL_unittest.cc and uses firstAllocation(), secondAllocation, and printResults() methods in CPL.

Automated? Yes

Preconditions for Test: Initialize a CPL object and give the object data equivalent to calling a handleCPL() in FileParser, and create a boolean vector pointer that points to a boolean vector.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls firstAllocation	CPL object, int quota, boolean vector pointer to a boolean vector	Should set the parties firstSeats and leftoverVotes.	Does the expected result.	
2	Calls secondAllocation	CPL object, int quota,	Should set the parties secondSeats.	Does the expected result.	

		boolean vector pointer to a boolean vector			
3	Checks that printResults() prints the correct terminal output	CPL object	See CPL_unittest.cc expect1 for expected result.	Does the expected result.	expect1 is in the protected part of CPLTest.

Postconditions for Test: Deletes the boolean vector.

—--

—--

Test Stage: Unit Test Date: 3/26/23
Test Case ID: CPL_testCalculateWinners() Name of Tester: Daniel
Test Description: Verifies functionality of printResults() of CPL class
Test Storage Location and Functions Used: Found in CPL_unittest.cc and uses calculateWinners() methods in CPL.
Automated? Yes
Preconditions for Test: Initialize a CPL object and give the object data equivalent to calling a handleCPL() in FileParser

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls calculateWinners()	CPL object	Should run firstAllocation(), secondAllocation(), and printResults().	Does the expected result.	
2	Checks that calculateWinners() prints the correct terminal output	CPL object	See CPL_unittest.cc expect1 for expected result.	Does the expected result.	expect1 is in the protected part of CPLTest.

Postconditions for Test: None.

—---

—---

Test Stage: Unit   Test Date: 3/26/23

Test Case ID: FileParser_testOpenFile()   Name of Tester: Daniel

Test Description: Verifies functionality of openFile() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses openFile() method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object, and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls openFile() on both election type .csvs and a failure case.	FileParser object, CPLTest1.csv, IRVTest.csv	Should return 0, 0, and 1, for each openFile().	Does the expected result.	

Postconditions for Test: None.

—---

—---

Test Stage: Unit   Test Date: 3/26/23

Test Case ID: FileParser_testIRVBallotToArray()   Name of Tester: Daniel

Test Description: Verifies functionality of ballotToArray() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses ballotToArray() method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object, and initialize several test ballots to check functionality of ballotToArray().

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls ballotToArray() on test ballots	FileParser object	Should convert the ballot string into a vector, which is then stored to test.	Does the expected result.	
2	Check the stored vectors and compare them to the expected values.	FileParser object	Should return {0,1,2,3,4}, {4,3,2,1,-1}, {0,4,1,3,-1}, {0,-1,-1,-1,-1}, and {3,4,-1,-1,-1} for each vector, respectively.	Does the expected result.	

Postconditions for Test: None.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: FileParser_testCPLReadBallots1() Name of Tester: Daniel

Test Description: Verifies functionality of readBallots() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses readBallots() method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object, and set up a CPL pointer that points to a CPL object as if it had been written to by getMetadata(), and that there are .csv files with the proper name to open.

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

1	Calls readBallots() on initialized CPL object	FileParser object, CPL pointer to CPL object. CPLTest1Ballots.csv	Should read the ballots into the CPL object.	Does the expected result.	
2	Check the CPL object's parties to see if they were modified correctly.	FileParser object, CPL pointer to CPL object.	Should return 3, 2, 0, 2, 1, 1, for each party, respectively.	Does the expected result.	

Postconditions for Test: Deletes the CPL object.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: FileParser_testHandleCPL1 Name of Tester: Daniel

Test Description: Verifies functionality of readBallots() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses handleCPL() method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls handleCPL() on the .csv with a preset date.	FileParser object, CPLTest1Handle.csv	Should return a CPL object that corresponds to the ballots parsed in	Does the expected result.	
2	Check the CPL object's parties to	FileParser object, CPL pointer to CPL	Should return 3, 2, 0, 2, 1, 1, for each	Does the expected result.	

	see if they were modified correctly.	object.	party, respectively.		

Postconditions for Test: Deletes the CPL object.

—---

—---

Test Stage: Unit Test Date: 3/26/23

Test Case ID: FileParser_testCPLReadBallots2() Name of Tester: Daniel

Test Description: Verifies functionality of readBallots() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses readBallots(CPL*) method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object, and set up a CPL pointer that points to a CPL object as if it had been written to by getMetadata() and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls readBallots() on initialized CPL object	FileParser object, CPL pointer to CPL object. CPLTest2Ballots.csv	Should read the ballots into the CPL object.	Does the expected result.	
2	Check the CPL object's parties to see if they were modified correctly.	FileParser object, CPL pointer to CPL object.	Should return 3, 2, 0, 2, 1, 1, for each party, respectively.	Does the expected result.	

Postconditions for Test: Deletes the CPL object.

—---

—---

Test Stage: Unit   Test Date: 3/26/23

Test Case ID: FileParser_testCPLHandleCPL2   Name of Tester: Daniel

Test Description: Verifies functionality of handleCPL() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses handleCPL() method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls handleCPL() on the .csv with a preset date.	FileParser object, CPLTest2Handle.cs v	Should return a CPL object that corresponds to the ballots parsed in	Does the expected result.	
2	Check the CPL object's parties to see if they were modified correctly.	FileParser object, CPL pointer to CPL object.	Should return 3, 2, 0, 2, 1, 1, for each party, respectively.	Does the expected result.	

Postconditions for Test: Deletes the CPL object.

—---

—---

Test Stage: Unit   Test Date: 3/26/23

Test Case ID: FileParser_testIRVReadBallots()   Name of Tester: Daniel

Test Description: Verifies functionality of readBallots() of FileParser class

Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses readBallots(IRV*) method in FileParser.

Automated? Yes

Preconditions for Test: Initialize a FileParser object, and set up a IRV pointer that points to a IRV object as if it had been written to by getMetadata() and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls readBallots() on initialized IRV object	FileParser object, IRV pointer to IRV object. IRVTestBallots.csv	Should read the ballots into the IRV object.	Does the expected result.	
2	Check the IRV object's ballot list head for the correct value.	FileParser object, IRV pointer to IRV object.	Should return 0, 0, 0, 2, 2, 3, for each linked list, respectively.	Does the expected result.	

Postconditions for Test: Deletes the IRV object.

—---

—---

Test Stage: Unit Test Date: 3/26/23
Test Case ID: FileParser_testHandleIRV Name of Tester: Daniel
Test Description: Verifies functionality of handleIRV() of FileParser class
Test Storage Location and Functions Used: Found in FileParser_unittest.cc and uses handleIRV(IRV) method in FileParser.
Automated? Yes
Preconditions for Test: Initialize a FileParser object and that there are .csv files with the proper name to open.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Calls handleCPL()	FileParser object,	Should read the	Does the expected	

	on the .csv with a preset date.	IRVTestHandle.csv	ballots into the IRV object.	result.	
2	Check the CPL object’s parties to see if they were modified correctly.	FileParser object,	Should return 0, 0, 0, 2, 2, 3, for each linked list, respectively.	Does the expected result.	

Postconditions for Test: Deletes the IRV object.

—---