Ballot Counting System (BCS):

Team 5

Project 2 (Agile) Tests

PBI 1: Multiple Input Files, Parse files and create ballot objects for OPL, ID: 1

By: Jason Phadnis
Inputs:

1. oplMFInput1A.txt and oplMFInput1B.txt which combined represent the same ballot info as oplTestInput.txt
2. oplMFInput2A.txt, oplMFInput2B.txt, oplMFInput2C.txt, and oplMFInput2D.txt combined are the same ballots as those in oplTestInput2.txt. The individual ballots have been distributed randomly.

Tests:

1. System test: run the program with both oplMFInput1A.txt and oplMFInput1B.txt as input. Try entering them in both orders.
2. System test: run the program with oplMFInput2A.txt, oplMFInput2B.txt, oplMFInput2C.txt, and oplMFInput2D.txt as input. Try entering them in different orders.

Output:

1. (Test 1): Pike and Foster should win from the D party and Borg from the R party. Press release field should be identical to that output when running oplTestInput.txt alone. Similarly all the audit files should be identical aside from the different ordering of ballots being assigned to candidates.
2. (Test 2): Volze (D), Berg (R), and Wartenberg (G), will always win. There is a three way tie for the last winner and will thus vary from run to run. These are: McClerg (R), Moll (I), and Pike (D).

Passed: Yes
Date: 4/21/21

PBI 2: Test multiple input files, test parse ballet for IR. ID: 2

By: Kobin J. Khadka

Inputs:

1. IRBallot.txt
2. IRBallot2.txt

Tests:

1. Test the function ParseBallotIR which counts the valid ballot and ignores the invalid ones.

Output:

1. Returned the total number of valid ballots..
2. Returned the valid ballot ID related to each ballot.

Passed: Yes
Date: 4/29/21

PBI 3: Test POBallot class with different properties inherited from OPLBallot class and verify it successfully creates a PO ballot object. ID: 3

By: Kobin J. Khadka

Inputs:

1. three different PO ballots that were created with a candidate list and three different ballot input strings.

Tests:

1. Test the PO ballot class constructor and verify if it creates the PO ballot with all the properties. Also make sure they are accessible.

Output:

1. Used getCandidate() to get the name of the candidate for all three ballots used as an input and returned Pike, Borg and Smith as expected.
2. Used getParty to get the name of the party for all three ballots and ‘D’, ‘R’, ‘I’ was returned as expected.
3. Used getID to get the ID number of the ballot and returned 1, 2, 3 as expected.

Passed: Yes
Date: 4/28/21

PBI 1: Multiple Input Files, Parse files and create ballot objects for IR, ID: 4

By: Brandon Schenck
Inputs:

1. irTestInput.txt
2. irTestMultiFile.txt
3. irTestMultiFile2.txt

Tests:

1. System test: run the system using irTestInput.txt, irTestMultiFile.txt, and irTestMultiFile2.txt as input. You can input these files in any order.

Output:

1. The final output should be Rosen as the winner along with a proper audit file and press release file.

Passed: Yes
Date: 4/18/2021

PBI 2: Invalidate IRV Ballots, Refactor IR ballot creation system to invalidate ballots , ID: 5

By: Brandon Schenck
Inputs:

1. irTestInput.txt

Tests:

1. System Test: input irTestInput.txt into the system.

Output:

1. The winner of the election will be Chou and when opening the Audit File you should only see ballots with id 1, 4, and 5 were created because ballots 2, 3, 6, and 7 were invalidated due to having <50% of the candidates ranked.

Passed: Yes
Date: 4/21/2021

PBI 3: PBI Process PO Ballots, Create POBallot objects, ID: 6

By: Jason Phadnis
Inputs:

1. poTestInput.txt
2. poMFInput1A.txt and poMFInput1B.txt which combined represent the same ballot info as poTestInput.txt

Tests:

1. System test: run the program with poTestInput.txt only as input. Uncomment the debugging for loop at the end of po() to confirm the results.
2. System test: run the program with both poMFInput1A.txt and poMFInput1B.txt as input. Try entering them in both orders. Uncomment the debugging for loop at the end of po() to confirm the results.

Output:

1. The following list in the console: Ballot number 1: Pike D, Ballot number 2: Pike D, Ballot number 3: Foster D, Ballot number 4: Jones R, Ballot number 5: Smith I, Ballot number 6: Borg R, Ballot number 7: Borg R, Ballot number 8: Pike D, Ballot number 9: Foster D
2. The following list in the console or this list with the first three moved to the end depending on the order in which you give the system the input files: Ballot number 1: Pike D, Ballot number 2: Pike D, Ballot number 3: Foster D, Ballot number 4: Jones R, Ballot number 5: Smith I, Ballot number 6: Borg R, Ballot number 7: Borg R, Ballot number 8: Pike D, Ballot number 9: Foster D

Passed: Yes

Date: 4/27/2021

PBI 1: Test Multiple Input Files, Parse ballots for OPL, ID: 7

Inputs:

OPLBallotDistribTest.txt

Tests:

1. Test the function parseBallotsForOPL. The file used to test this function has been provided by us.
2. Test the function createAndDistributeOPLBallots inside of parseBallotsForOPL. The file used to test this function has been provided by us.

Output:

1. Used .size() on the arraylist of ballots to count the amount of ballots parseBallotsForOPL processed. Since there are 18 ballots in OPLBallotDistribTest.txt, the output is 18
2. Since parseBallotsForOPL contains the function createAndDistributeOPLBallots(), getPartyVoteCount() is used to count the number of votes each party gets. This should

come out as D gets 6 votes total, R gets 6 votes total, G gets 4 votes total, and I gets 2 votes total.

By Gavin Huang
Passed: Yes
Date: 4/27/21

Project 1 (Waterfall) Tests

Test Stage: Unit	Test Date: 3/11/21
Test Case ID#: OPLBallot_Creation	Name of Testers: Jason

Test Description:
Test creating OPLBallot objects. TestOPLBallot.java has the test code which can be run using TestRunner.java.

Automated: yes
Results: Pass

Preconditions for Test:
The ballotString argument for the constructor is a line from the input file in the form ,,1,,, where the 1 represents the candidate that this ballot is voting for. There should be no characters other than "," and "1" and there should be only one "1" in this string. The candidateList argument is in the form of [name],[party] e.g. Pike,D. There should be only letter characters for the party and no elements should be missing.

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

1	Create an OPL ballot where the first candidate listed is chosen	ballotString is given "1,,,,," and the candidateList is {"Pike,D", "Foster,D", "Deutsch,R", "Borg,R", "Jones,R", "Smith,I"}	OPLBallot's Candidate name: Pike Party: D Id number: 1	OPLBallot's Candidate name: Pike Party: D Id number: 1	
2	Create an OPL ballot where a middle candidate listed is chosen	ballotString is given ",,,1,," and the candidateList is {"Pike,D", "Foster,D", "Deutsch,R", "Borg,R", "Jones,R", "Smith,I"}	OPLBallot's Candidate name: Borg Party: R Id number: 2	OPLBallot's Candidate name: Borg Party: R Id number: 2	
3	Create an OPL ballot where the last candidate listed is chosen	ballotString is given ",,,,,1" and the candidateList is {"Pike,D", "Foster,D", "Deutsch,R", "Borg,R", "Jones,R", "Smith,I"}	OPLBallot's Candidate name: Smith Party: I Id number: 3	OPLBallot's Candidate name: Smith Party: I Id number: 3	

Post condition(s) for Test:
An OPLBallot is created with the appropriate candidate name, candidate party, and ballot id number set.

Test Stage: Unit	Test Date: 3/11/21

Test Case ID#: BCS_CoinFlip	Name of Testers: Jason

Test Description:
Make sure that coinFlip() not only returns only 0 or 1, but that the total amount of each is about equal to each other. If they are, then a fair coin toss is simulated correctly. This test is testCoinFlip() located in TestBCS.java which can be run through TestRunner.java.

Automated: no
Results: Pass

Preconditions for Test:
None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Simulate a coin flip 100 times. I also see if the first 5 flips of every test run are different (i.e. if the seed is different between runs)	None	zeroCount + oneCount = 100 and zeroCount and oneCount should both be about 50. The first 5 coin tosses are not the same between all test runs.	The greatest deviation I got was 34 of one and 66 of the other. Most tests however were around 50 += 5 The first 5 coin tosses are not the same between all test runs.	I ran this test 10 times to verify if the two counts were consistently about 50 each.

Post condition(s) for Test:
Should produce only zeros and ones. Total of zeros and ones should be 100 and they should be about 50/50 each.

Test Stage: Unit	Test Date: 3/11/21

Test Case ID#: BCS_DateTime	Name of Testers: Jason

Test Description:
Test to see if BCS.java's getDateTime() is giving a string of the form "[currentDate]_[currentTime]". This test is testDateTime() located in TestBCS.java which can be run through TestRunner.java.

Automated: no
Results: Pass

Preconditions for Test:
None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call getDateTime() and ensure the time is correct	None	Two print statements. The fist is the date, and the second is the time.	Two print statements. The fist is the date, and the second is the time.	The print statements get the result by splitting the string from getDateTime on the "_"

Post condition(s) for Test:

Test Stage: Unit	Test Date: 3/12/21
Test Case ID#: BCS_AuditFileCreation	Name of Testers: Jason

Test Description:
Try creating the audit file for the different election types. This test function is called testCreateAuditFile() and it is in BCS.java.

Automated: no
Results: Pass

Preconditions for Test:
The electionType private variable in BCS should be set.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Try creating an audit file for an IR election	electionType = 1	An audit file called IR[date][time]AuditFile.txt is created. The file is empty.	An audit file called IR[date][time]AuditFile.txt is created. The file is empty.	
2	Try creating an audit file for an OPL election	electionType = 2	An audit file called OPL[date][time]AuditFile.txt is created. The file is empty.	An audit file called OPL[date][time]AuditFile.txt is created. The file is empty.	
3	Try creating an audit file with electionType != 1 or 2	electionType = 0	A print statement to the console saying: Improper election type of 0	A print statement to the console saying: Improper election type of 0	

Post condition(s) for Test:
Two text files should be in the same directory as BCS.java after running the tests (one for each of the first two steps).

Test Stage: Unit	Test Date: 3/12/21
Test Case ID#: BCS_CreateParties	Name of Testers: Jason

Test Description:

Create Party objects for use in OPL and create and add the appropriate Candidate objects. The test method is in TestBCS.java and it's called testPartyCreation(). This test method can be run with TestRunner.java.

Automated: yes
Results: Pass

Preconditions for Test:
Must be given an empty ArrayList of Party objects and an array of Strings representing the candidates. Each string should be in the form "candidateName,Party".

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a party using an empty candidate array string.	String[] emptyCandidateInfo = {}; ArrayList<Party> parties = new ArrayList<Party>();	No print statements and parties.size() == 0	No print statements and parties.size() == 0	
2	Create a party using a candidate array string with members of only one party.	String[] onePartyCandidateInfo = {"Smith,I", "Jackson,I", "Dickson,I", "Sarah,I"}; parties = new ArrayList<Party>();	parties.size() == 1 and Print statements: The candidates in the I party: Smith, Jackson, Dickson, Sarah,	parties.size() == 1 and Print statements: The candidates in the I party: Smith, Jackson, Dickson, Sarah,	
3	Create parties using a candidate array string with members of multiple parties intermixed.	tring[] candidateInfo = {"Smith,I", "Jones,R", "Dickson,I", "Rosen,D", "Phil,D"}; parties = new	parties.size() == 3 and Print statements: The candidates in the I party: Smith,	parties.size() == 3 and Print statements: The candidates in the I party: Smith,	The order of the candidates within each party and of the parties themselves are expected

		ArrayList<Party>();	Dickson, The candidates in the R party: Jones, The candidates in the D party: Rosen, Phil,	Dickson, The candidates in the R party: Jones, The candidates in the D party: Rosen, Phil,	to be in the given order as well.

Post condition(s) for Test:
The ArrayList passed in as an argument to createParties() should contain the Party objects created with their candidate lists filled in with one of each of the appropriate candidate objects. There should be one party object for each party in the input string.

Test Stage: Unit	Test Date: 3/12/21
Test Case ID#: BCS_OPLBallotDistribution	Name of Testers: Jason

Test Description:
Test creating and distributing OPLBallot objects to the appropriate candidates. This includes checking that the total votes for each candidate and for each party are correct. This is tested using the testOPLBallotDistribution() function which is within BCS.java.

Automated: yes
Results: Pass

Preconditions for Test:
An array of OPLballots with length equal to the number of ballots in the input file is passed into the function. Furthermore, the file reader's next line to read is the first ballot in the input field. Another argument is a string array of the candidates and their parties in the form "[name],[party]", e.g. "Pike,D". The final argument is an ArrayList of Party objects with all the appropriate candidates.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Use a test file of	The	Party totals in	Party totals in	

	appropriate ballot info that aligns with an example party set.	candidates are: "Volze,D", "Moll,I", "Wartenberg, G", "Foster,D", "Berg,R", "McClerg,R", "Morey,G", "Pike,D", "Grolnie,R" A correspondin g ballot input file called OPLBallotDi stribTest.txt	order D, R, G, I: 6, 6, 4, 2 Candidate total votes in order Volze, Foster, Pike, Berg, McClerg, Grolnie, Wartenberg, Morey, Moll: 3, 1, 2, 3, 2, 1, 3, 1, 2	order D, R, G, I: 6, 6, 4, 2 Candidate total votes in order Volze, Foster, Pike, Berg, McClerg, Grolnie, Wartenberg, Morey, Moll: 3, 1, 2, 3, 2, 1, 3, 1, 2	

Post condition(s) for Test:
The parties and individual candidates have appropriate vote counts given the ballots.

Test Stage: Unit	Test Date: 3/12/2021
Test Case ID# IRBallot_testIRBallotCreation	Name of Testers: Brandon

Test Description:
This test creates irBallot objects to check to see if the IRBallot constructor is working properly. These tests are in the TestIRBallot.java file and the method being used is testIRBallotCreation.

Automated: yes
Results: Pass

Preconditions for Test: Have a candidate names array a candidate parties array created. The method creates four ballots of IRBallot type and passes into these objects a string depicting the ballot rankings, an array of candidate names, an array of parties for each candidate, and an id number.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test when each person has a vote with ranking in order.	String irBallotString1 = "1,2,3,4";	‘Rosen’: name ‘D’: party 1 : id	‘Rosen’: name ‘D’: party 1 : id	
2	Test when each person has a vote with rankings not in numerical order.	String irBallotString2 = "2,3,4,1";	‘Royce’: name ‘L’: party 2 : id	‘Royce’: name ‘L’: party 2 : id	
3	Test when there is a person not ranked	String irBallotString3 = "3,,2,1";	‘Royce’: name ‘L’: party 3 : id	‘Royce’: name ‘L’: party 3 : id	
4	Test when only one person is ranked	String irBallotString4 = ",1,,";	‘Kleinberg’: name ‘R’: party 4 : id	‘Kleinberg’: name ‘R’: party 4 : id	

Post condition(s) for Test:
The asserts should all have equal expected and resulting values. This test should indicate that it passed.

Test Stage: Unit	Test Date: 3/13/21
Test Case ID#: BCS_distributeSeatsToParties	Name of Testers: Jason

Test Description:
Test to see if the individual parties’ numSeatsToDistribute variables are set appropriately given the number of votes. This test is run via the testDistributeSeatsToParties() function in BCS.java.

Automated: no
Results: Pass

Preconditions for Test:
The number of ballots and number of seats are set correctly by the input file. The party objects are created and has party vote counts appropriately based on the input file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test when there is an even seat distribution	D (6): Volze = 3, Foster = 1, Pike = 2 R (6): Berg = 3, McClerg = 2, Grolnie = 1 G (4): Wartenberg = 3, Morey = 1 I (2): Moll = 2 numSeats = 3	Num seats by party: D: 1 R: 1 G: 1 I: 0	Num seats by party: D: 1 R: 1 G: 1 I: 0	
2	Test when there is an uneven seat distribution	D (6): Volze = 3, Foster = 1, Pike = 2 R (6): Berg = 3, McClerg = 2, Grolnie = 1 G (4): Wartenberg = 3, Morey = 1 I (2): Moll = 2 numSeats = 4	Num seats by party: D: 1 or 2 R: 1 G: 1 or 2 I: 0 or 1 D+G+I = 3	Num seats by party: D: 1 or 2 R: 1 G: 1 or 2 I: 0 or 1 D+G+I = 3	
3	Test when there is a larger even seat distribution	D (6): Volze = 3, Foster = 1, Pike = 2 R (6): Berg = 3, McClerg = 2, Grolnie = 1 G (4): Wartenberg = 3, Morey = 1 I (2): Moll = 2 numSeats = 6	Num seats by party: D: 2 R: 2 G: 1 I: 1	Num seats by party: D: 2 R: 2 G: 1 I: 1	

4	Test when the seat number is equal to the number of candidates	D (6): Volze = 3, Foster = 1, Pike = 2 R (6): Berg = 3, McClerg = 2, Grolnie = 1 G (4): Wartenberg = 3, Morey = 1 I (2): Moll = 2 numSeats = 9	Num seats by party: D: 3 R: 3 G: 2 I: 1	Num seats by party: D: 3 R: 3 G: 2 I: 1	
5	Test when the number of seats is 0	D (6): Volze = 3, Foster = 1, Pike = 2 R (6): Berg = 3, McClerg = 2, Grolnie = 1 G (4): Wartenberg = 3, Morey = 1 I (2): Moll = 2 numSeats = 0	Error message in the console	Error message in the console	

Post condition(s) for Test:
Each party has the correct number for their numSeatsToDistribute variable.

Test Stage: System	Test Date: 3/13/21
Test Case ID#: BCS_RunningOPL	Name of Testers: Jason

Test Description:
Testing the system with an OPL election. Run the main program (BCS.java) using the file "oplTestInput.txt" located in the testing folder.

Automated: no
Results: Pass

Preconditions for Test:
The input file, oplTestInput.txt, should have no errors in it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the main program with an input file for an OPL election.	See oplTestInput.txt for all the data. There are 9 ballots and 3 seats available.	Pike (D), Foster (D), and Borg(R) will always win.	Pike (D), Foster (D), and Borg(R) will always win.	

Post condition(s) for Test:
A press release file and an audit file will be created based on the run of the program with that particular input file. Some results will also appear in the console.

Test Stage: System	Test Date: 3/13/21
Test Case ID#: BCS_RunningOPL2	Name of Testers: Jason

Test Description:
Testing the system with an OPL election. A number of seats and vote distribution is chosen such that at least one coin toss will be involved to determine the final result. Run the main program (BCS.java) using the file “oplTestInput2.txt” located in the testing folder.

Automated: no
Results: Pass

Preconditions for Test:
The input file, oplTestInput2.txt, should have no errors in it.

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

1	Run the main program with an input file for an OPL election.	See oplTestInput2 .txt for all the data. There are 18 ballots and 4 seats available.	Volze (D), Berg (R), and Wartenberg (G), will always win. There is a three way tie for the last winner and will thus vary from run to run. These are: McClerg (R), Moll (I), and Pike (D)	Volze (D), Berg (R), and Wartenberg (G), will always win. There is a three way tie for the last winner and will thus vary from run to run. These are: McClerg (R), Moll (I), and Pike (D)	I ran the program with this file multiple times to ensure the different results based on the coin toss.

Post condition(s) for Test:
A press release file and an audit file will be created based on the run of the program with that particular input file. Some results will also appear in the console.

Test Stage: Unit	Test Date: 03/13/21
Test Case ID#: Party_testPCreation	Name of Testers: Gavin

Test Description:

The purpose of this test is to evaluate and analyze the distribution of seats that are calculated from the OPL diagram towards all the candidates. This test is run via the distributeSeatsToCandidates() in party.java and other supporting functions.

(indicate where you are storing the tests and the name of the method being used)
Automated: no
Results: Pass

Preconditions for Test:

The number of ballots and number of seats are set correctly by the input file. The party objects are created and has party vote counts appropriately based on the input file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test to increment the party vote count for each party	D (1): Caillou, Rosie R(1): Elmo, CookieMonster C(1): Arthur DW F(1): George, ManInTheYellowHat	Num Seats by Party D: 2 R: 2 C: 2 F: 2	Num Seats by Party D: 2 R: 2 C: 2 F: 2	
2	Test to implement the numbers of seats to distribute	D (1): Caillou, Rosie R(1): Elmo, CookieMonster C(0): Arthur DW F(0): George, ManInTheYellowHat	Seats Distributed Available: D: 1 R: 1 C: 0 F: 0	Seats Distributed Available: D: 1 R: 1 C: 0 F: 0	
3	Test to distribute the seats to candidates based on the number of seats	D (1): Caillou, Rosie R(1): Elmo, CookieMonster C(0): Arthur DW F(0): George,	Seats Distributed: D: 1 R: 1 C: 0 F: 0	Seats Distributed: D: 1 R: 1 C: 0 F: 0	

		ManInTheYellowHat			

Post condition(s) for Test:

Each candidate who have won a seat will correctly be distributed

---

Test Stage: Unit	Test Date: 03/13/21
Test Case ID#: Report	Name of Testers: Gavin

Test Description:
The purpose of this test is to evaluate and analyze the printing of the election report and creating a press release file. This test is run through the functions of printReport() and generateReportFile().

(indicate where you are storing the tests and the name of the method being used)
Automated: no
Results: Pass

Preconditions for Test:

The report object has been created with inputted values to replicate the IR and OPL election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test for printing the results in the terminal	IR: Num of Candidate: 3 Num Of Ballot: 1 Winner: Uvuvwevwevwe Onyetenyevwe	Below are the results of the IR election The number of candidates in this election was 3 The number of ballots in this election	Below are the results of the IR election The number of candidates in this election was 3 The number of ballots in this election	

		Ugwemubwem Ossas OPL: Num of Candidate: 3 Num Of Ballot: 1 Winner who won seat(s): Hubert Blaine Wolfeschlegelsteinhausenbergerdorff Sr	was 1 And the winning candidate of the Instant Runoff election is: Uvuvwevwevwe Onyetenyevwe Ugwemubwem Ossas Below are the results of the OPL election The number of candidates in this election was 3 The number of ballots in this election was 1 And the candidates who won seats in this Open Party Listing election are: Hubert Blaine Wolfeschlegelsteinhausenbergerdorff Sr	was 1 And the winning candidate of the Instant Runoff election is: Uvuvwevwevwe Onyetenyevwe Ugwemubwem Ossas Below are the results of the OPL election The number of candidates in this election was 3 The number of ballots in this election was 1 And the candidates who won seats in this Open Party Listing election are: Hubert Blaine Wolfeschlegelsteinhausenbergerdorff Sr	
2	Test for generating a press release file	IR: Date 03-13-21 CandidateStandings: IR Press Release Info	Press release file named: 03-13-21PressRelease.txt created successfully Press release file named:	Press release file named: 03-13-21PressRelease.txt created successfully Press release file named:	

		OPL: Date 01-01-01 CandidateStandings: OPL Press Release Info	01-01-01PressRelease.txt created successfully	01-01-01PressRelease.txt created successfully	

Post condition(s) for Test:

The results should print the results from the inputted values and create a press release file from the result object for IR and OPL

Test Stage: Unit	Test Date: 3/14/21
Test Case ID#: CandidateTest	Name of Testers: Kobin

Test Description: This test is done on candidate class and the methods within the candidate class to verify they all work properly. Test file is named as TestCandidate.java. There are three methods used: testCandidateCreation, testVoteCount and the ballotList .

Automated: Yes
Results: Pass

Preconditions for Test: Candidate object is created in each method with their name and the party they belong to. Also, the parties char array and candidate name array is created and used for the ballotList method.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test if the name of the candidate is returned correctly.	Candidate candidate = new Candidate("Pike", 'D');	Pike	Pike	
2	Test if the party name is returned correctly.	Candidate candidate = new Candidate("Pike", 'R');	D	D	
3	Test if the new vote is added to the candidate	Candidate candidate = new Candidate("Pike", 'R');	2	2	
4	Test if vote is counted for candidate	Candidate candidate = new Candidate("Pike", 'R');	2	2	
5	Test if new ballot are added to the ballot list by looking at the ballot ID	IRBallot ir = new IRBallot(irBallotString1, parties, candidateNames, 100); IRBallot ir2 = new IRBallot(irBallotString2, parties, candidateNames, 101);	100	100	

		IRBallot ir3 = new IRBallot(irBallotString3, parties, candidateNames, 102); IRBallot ir4 = new IRBallot(irBallotString4, parties, candidateNames, 103);			
6	Test if the ballot size is increasing when new ballot are added	IRBallot ir = new IRBallot(irBallotString1, parties, candidateNames, 100); IRBallot ir2 = new IRBallot(irBallotString2, parties, candidateNames, 101); IRBallot ir3 = new IRBallot(irBallotString3, parties, candidateNames, 102); IRBallot ir4 = new IRBallot(irBallotString4, parties, candidateNames, 103);	2	2	

Post condition(s): Candidate and their party will be created successfully. Also, the system will be able to add new ballots in the ballot list , and get votes from the ballot and add it to the candidate.

Test Stage: System	Test Date: 3/14/21
Test Case ID#: BCS_RunningOPLTiedParties	Name of Testers: Jason

Test Description:
Check to see if two parties' total votes are tied for getting an extra seat that either of them could get it. Run the main program (BCS.java) using the file "oplTestInputTiedParties.txt" located in the testing folder.

Automated: no
Results: Pass

Preconditions for Test:
Need properly formatted oplTestInputTiedParties.txt file in the same directory as the BCS program.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the program multiple times and see if the D party gets the third seat in some cases and the R party in others.	See oplTestInput TiedParties.tx t for all the data. There are 10 ballots and 3 seats available.	Pike (D) and Borg ® should always win. Then either a member of the R party or the D party will get the third seat. Pike and Borg are the only pens with votes so if the R party gets the third	Pike (D) and Borg ® should always win. Then either a member of the R party or the D party will get the third seat. Pike and Borg are the only pens with votes so if the R party gets the third	I ran this test multiple times to confirm that the D party wins the third some times and the R party the other times.

			seat either of Deutsch and Jones can win. Foster wins if D gets the third seat.	seat either of Deutsch and Jones can win. Foster wins if D gets the third seat.	

Post condition(s) for Test:
A press release file and an audit file will be created based on the run of the program with that particular input file. Some results will also appear in the console.

Test Stage: System	Test Date: 3/14/21
Test Case ID#: BCS_RunningOPLMultipleSeats	Name of Testers: Jason

Test Description:
Make sure that a party can get more than one seat in the first round of seat allocation. Run the main program (BCS.java) using the file “oplTestInputMultipleSeats.txt” located in the testing folder.

Automated: no
Results: Pass

Preconditions for Test:
Need properly formatted oplTestInputMultipleSeats.txt file in the same directory as the BCS program.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to make sure the D party gets two seats in the first round of allocation.	See oplTestInput MultipleSeats .txt for all the data.	Audit file indicates that two seats are given to the D party in the first allocation	Audit file indicates that two seats are given to the D party in the first allocation	

		There are 9 ballots and 3 seats available.	round.	round.	

Post condition(s) for Test:
A press release file and an audit file will be created based on the run of the program with that particular input file. Some results will also appear in the console.

Test Stage: Unit	Test Date: 3/14/2021
Test Case ID#: IRBallot_testSetNewTopPick	Name of Testers: Brandon

Test Description:
This tests if an irBallot is still valid and if it needs a new top pick as well as testing when a ballot needs to be thrown out. The test is being stored in TestIRBallot.java and the method being used is testSetNewTopPick().

Automated: yes
Results: Pass

Preconditions for Test: This test needs candidate objects to be created and for those candidate objects to be separated into lists of available candidates and eliminated candidates. This test also needs IRBallot objects to be created. After these are created we then test

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test to see if first ballot is valid	String irBallotString 1 = "1,2,3,4"; IRBallot irBallot1 = new IRBallot(irBa llotString1,ca	true	true	

		ndidateParties,candidateNames,id);			
2	Test to see if second ballot is valid	String irBallotString2 = "2,3,4,1"; IRBallot irBallot2 = new IRBallot(irBallotString2,candidateParties,candidateNames,id);	true	true	
3	Test to see if third ballot is valid	String irBallotString3 = "3,2,1,"; IRBallot irBallot3 = new IRBallot(irBallotString3,candidateParties,candidateNames,id);	true	true	
4	Test to see if fourth ballot is valid	String irBallotString4 = ",1,,"; IRBallot irBallot4 = new IRBallot(irBallotString4,candidateParties,candidateNames,id);	false	false	

Post condition(s) for Test:

All of the asserts should pass and the ballots that returned true and needed a new top pick would be set or stay on their current top pick and the ballots that returned false would be thrown out.

Test Stage: Unit	Test Date: 3/14/2021
Test Case ID#: BCS_testGetIRCandidateParties	Name of Testers: Brandon

Test Description:
Tests the function getIRCandidateParties by taking in a string array of candidate names with their parties and then returning an array of characters with just the parties of the candidates. This test is stored in BCS.java and is called testGetIRCandidateParties().

(indicate where you are storing the tests and the name of the method being used)
Automated: yes
Results: Pass

Preconditions for Test: The preconditions for this test is an array of strings that consist of the candidate name and party.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test array of candidates with party	String[] candidatesWithParty = {"Rosen (D)", "Kleinberg (R)", "Chou (I)", "Royce(L)"};	{'D','R','I','L'}	{'D','R','I','L'}	

Post condition(s) for Test:
The post condition for this test is the creation of a new character array containing the parties of each candidate.

Test Stage: Unit	Test Date: 3/14/2021
Test Case ID#:BCS_testGenerateIRReport	Name of Testers: Brandon

Test Description:
This test tests the creation of a press release file and results printing to the screen. This test is located in BCS.java and the method is called testGenerateIRReport().

Automated: yes
Results: Pass

Preconditions for Test: This test requires the creation of a list of availableCandidates, a list of eliminated candidates, the winner of the election, the total number of ballots, and the total number of candidates.

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

1	Tests the generateIRReport() function	int numCandidates = 2;
		int numBallots = 10;
		Candidate rosen = new Candidate("Rosen",'D');
		Candidate chou = new Candidate("Chou",'R');
		
		ArrayList<Candidate> availableCandidates = new ArrayList<>();
		
		availableCan
			A report file to be generated and results printed to the screen.	A report file is generated and results are printed to the screen.	

		didates.add(rosen);
		
		ArrayList<Candidate> eliminatedCandidates = new ArrayList<>();
		
		eliminatedCandidates.add(chou);			

Post condition(s) for Test:
A report file is created and results of the election are printed to the screen.

Test Stage: Unit	Test Date: 3/14/2021
Test Case ID#: BCS_testPrintResults	Name of Testers: Brandon

Test Description:
This test writes the results of the election to the audit file. This test is stored in the BCS.java file and the method name is testPrintResults().

Automated: yes
Results: Pass

Preconditions for Test:
This test creates an audit file, specifies the type of election, number of candidates, number of ballots, an arraylist of available candidates, an arraylist of eliminated candidates, and a winner.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test printResults()	Candidate rosen = new Candidate("R	An audit file is created and results are	An audit file is created and results are	

		osen",'D');
		Candidate chou = new Candidate("Chou",'R');
		
		ArrayList<Candidate> availableCandidates = new ArrayList<>();
		
		availableCandidates.add(rosen);
		
		ArrayList<Candidate> eliminatedCandidates = new ArrayList<>();
		
		eliminatedCandidates.add(chou);
			written to it.	written to it.	

Post condition(s) for Test:
This test generates an audit file with results written to it.

Test Stage: Unit	Test Date: 3/14/2021
Test Case ID#: BCS_testDistributeIRVotes	Name of Testers: Brandon

Test Description:
Tests the function, distributeIRVotes(). This method is located in BCS.java and the method being used is testDistributeIRVotes().

(indicate where you are storing the tests and the name of the method being used)

Automated: yes
Results: Pass

Preconditions for Test:
The preconditions for this test are the creation of ballots and candidates that are then stored in lists and passed into the function that is being tested.

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

1	Test distributeIRVotes()	ArrayList<IRBallot> irBallotList = new ArrayList<>();
		IRBallot irBallot1 = new IRBallot(irBallotString1,candidateParties,candidateNames,id);
		
		irBallotList.add(irBallot1);
		id++;
		IRBallot irBallot2 = new IRBallot(irBallotString2,candidateParties,candidateNames,id);
		
		irBallotList.add(irBallot2);
		id++;
		IRBallot irBallot3 = new IRBallot(irBallotString3,ca
			The votes are distributed to each candidate and an audit file is created.	The votes are distributed to each candidate and an audit file is created.	

		ndidateParties,candidateNames,id);
		
		irBallotList.add(irBallot3);
		id++;
		IRBallot irBallot4 = new IRBallot(irBallotString4,candidateParties,candidateNames,id);
		
		irBallotList.add(irBallot4);
		id++;
		
		ArrayList<Candidate> availableCandidates = new ArrayList<>();
		Candidate rosen = new Candidate("Rosen",'D');
		
		availableCandidates.add(rosen);
		Candidate kleinberg = new Candidate("Kleinberg",'R');
		
		availableCandidates.add(kleinberg);
		Candidate chou = new			

		Candidate("Chou",'I');
		
		availableCandidates.add(chou);
		Candidate royce = new Candidate("Royce",'L');
		
		availableCandidates.add(royce);			

Post condition(s) for Test:
An audit file is created with votes distributed to candidates.

Test Stage: System	Test Date: 3/14/2021
Test Case ID#: BCS_testIR	Name of Testers: Brandon

Test Description:
This test runs IR on a sample ir test file called irTestInput.txt. This file is stored in the test folder.

Automated: no
Results: Pass

Preconditions for Test: start the program and input the irTestInput.txt file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check the audit file	irTestInput.txt	An audit file with a history of what happened in the election	An audit file with a history of what happened in the election	

2	Check the press release file	irTestInput.txt	A press release file with information about the election that can be shared with the media	A press release file with information about the election that can be shared with the media	
3	Check results printed to the screen	irTestInput.txt	Results of the election including who the winner is.	Results of the election including who the winner is.	

Post condition(s) for Test:
You will have results printed to the screen and an audit file and press release file generated with information about the election and what happened throughout the running of the voting algorithms.

Test Stage: System	Test Date: 3/14/2021
Test Case ID#: BCS_testIR2	Name of Testers: Brandon

Test Description:
This test runs an election with only two candidates and creates an audit file and result file for the results of the election.

Automated: No
Results: Pass

Preconditions for Test:
The BCS system needs to be run in the terminal and you need to use the test file irTestInput2.txt.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test irTestInput2.txt	irTestInput2.t	An audit file	An audit file	

		xt	and press release file are created and correct results are written to it.	and press release file are created and correct results are written to it.	

Post condition(s) for Test:
This test generates an audit file with the history of the election and the results file contains the results of the election and other helpful information about the election.

Test Stage: System	Test Date: 3/14/2021
Test Case ID#: BCS_testIR3	Name of Testers: Brandon

Test Description:
This test runs an election with seven candidates and creates an audit file and result file for the results of the election. This test was created to show the handling of ties and thrown out ballots.

Automated: No
Results: Pass

Preconditions for Test:
This test needs the BCS system to run and irTestInput3.txt to run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test irTestInput3.txt	irTestInput3.txt	An audit file and press release file are created and correct results are written to it.	An audit file and press release file are created and correct results are written to it.	

Post condition(s) for Test:
This test generates an audit file with the history of the election and the results file contains the results of the election and other helpful information about the election.
