Test Plan

Group: I

Names: Nasser Alsuhaimi, Ryan Huffman, Leticia Banuelos.

**Introduction**

This serves as the plan for testing the software of the digital implementation of the *Illuminati* board game, as well as the reporting of test results.

**For all of the test cases outlined in this document, all other outcomes that are not included in this document are failure cases and the test is considered unsuccessful.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 100.1 | Checks shuffle Illuminati cards   1. Randomly run shuffle on Illuminati cards to check if cards come out differently each time. | Game randomly draws an Illuminati card. |  |
| 100.2 | Checks if drawn Illuminati card is removed from deck.   1. Checks the number or remaining Illuminati cards in deck. | Illuminati card player draws is removed from Illuminati card deck. |  |
| 100.3 | Checks if each player gets an Illuminati card.   1. Accesses player Illuminati card. | Each player draws a different Illuminati card. |  |
| 100.4 | Checks if player draws “The UFOs”.   1. Accesses player Illuminati card information. | Player can select a win condition. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 101.1 | Checks shuffle group and special cards   1. Shuffle deck and draw a card to check that every time a card is drawn its different. | Game randomly shuffles and draws a group or special card. |  |
| 101.2 | Checks if card is a special card   1. Access card’s information | Game puts special card back in the deck. |  |
| 101.3 | Checks if card is a group card   1. Access card’s information | Game puts group card in uncontrolled group |  |
| 101.4 | Checks that drawn group card is removed from deck.   1. Checks the number of remaining group and special cards in deck. | Drawn group card is removed from deck. |  |
| 101.5 | Checks uncontrolled groups to make sure there are four groups.   1. Display uncontrolled group cards. | Game displays uncontrolled group cards information. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 102.1 | Checks if the roll is recorded.   1. The game should record every player roll. | Player roll is saved. |  |
| 102.2 | Check if the player who have the highest is playing first.   1. If the roll number for the player is higher than the other players roll. 2. Player should play first. | The first player to play is the person with the highest roll |  |
| 102.3 | Check if all players roll the dice.   1. Check If the “playerRoll” function was executed. 2. If not the execute the “playerRoll” function | All players should roll the dice |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 103.1 | Check if the group is a group with income.   1. Check if in that card class have an income or not 2. If not then dont give money | Only groups with income, should get the money |  |
| 103.2 | Check if the given money to the group is drawn from the bank.   1. Check if the drawn money id from the bank. 2. If not the take it from the bank. | The bank should give the groups the income |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 104.1 | Check if the given card “ Group” is faced up in the center. | The Group card should be placed faced up in the center with the other uncontrolled groups. |  |
| 104.2 | Check if the given card “special” is faced up or down based on the player happens. | The special card should be placed faced up or down based on the player choose. |  |
| 104.3 | Check if the card given is drawn from the deck. If no | Deck should remove the card given. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 105.1 | Check if the player has that amount of money. if NO | A prompt should appear stating the amount entered is invalid |  |
| 105.2 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 106.1 | Check if has at least one special card. If No: | A prompt should appear stating the player cannot use Specialty because there is none. |  |
| 106.2 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |
| 106.3 | Check if the Specialty card has been executed. If yes | Discard the specialty card. |  |
| 106.4 | Check if the Specialty card has been executed. If no | Execute the Specialty card then discard it. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 107.1 | Check if has at least one special card. If No: | A prompt should appear stating that the player cannot drop a group because they don’t have a group. |  |
| 107.2 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |
| 107.3 | Check if the Group has children. If yes | The application removes the Group’s children from the player’s Power Structure and returns it to the uncontrolled area. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 108.1 | Check if the player has at least two Groups. If no | A prompt should appear stating that player cannot transfer the money, because he has less than two groups. |  |
| 108.2 | Check if the selected group to transfer from is not the same group as the transfer to. If yes. | A prompt should appear stating that player cannot transfer the money to the same group that is transfer from. |  |
| 108.3 | Check if the selected group (transfer from) is adjacent to the group (transfer to). If no | A prompt should appear stating that in order to complete the transfer the two groups should be adjacent. |  |
| 108.4 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |
| 108.5 | Check if the group (transfer from) has that amount of money. if NO | A prompt should appear stating the amount entered is invalid |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 109.1 | Check if the player has at least one group. if no. | A prompt should appear stating the player don’t have any group. |  |
| 109.2 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 110.1 | Check if has at least one special card. If No | A prompt should appear stating the player cannot Give a Card because there is no special card. |  |
| 110.2 | If the player selects “No” when the application asked for confirmation. | A prompt should appear stating that the action has been canceled |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 200.1 | Check if the attacking player is prompted for the attack type.   1. Begin an attack | The prompt should appear when the attack begins. It should contain three options, “Control”, “Neutralize”, and “Destroy |  |
| 200.2 | Check if GamePlay records the attack type correctly.   1. Select an attack type on the prompt 2. Check the attack type stored by GamePlay 3. Repeat steps 1 and 2 for each attack type to make sure all of the buttons work | The attack type recorded by GamePlay should match the attack type pressed on the prompt. |  |
| 200.3 | Check if the attacking player is prompted to select the attacking group.   1. Should occur automatically after selecting attack type (200.2) | The prompt should appear after selecting the attack type. |  |
| 200.4 | Check if GamePlay records the selected group as the attacking group.   1. Select a group 2. Check the group stored by GamePlay 3. Repeat steps 1 and 2 for multiple groups, including both GroupCards and IlluminatiCards | The group recorded by GamePlay matches the selected group in every test. |  |
| 200.4.1 | Check if an error prompt occurs if a group with no power is selected   1. Following 200.4, instead select a group with no power | A prompt should appear stating the attacking group is invalid because it has no power |  |
| 200.4.2 | Check if an error prompt occurs if a group with no open control arrows is selected when the attack type is control   1. Set the attack type to control 2. Select an attacking group with no open control arrows | A prompt should appear stating the attacking group is invalid because it has no open control arrows. |  |
| 200.4.3 | Check if an error prompt occurs if a group’s attack counter is zero (meaning it has already attacked as many times as it could this turn)   1. Set a group’s attack counter to zero 2. Select that group as the attacking group | A prompt should appear stating the attacking group is invalid because it has already used its attacks this turn. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 201.1 | Check if the attacking player is prompted for the group to attack.   1. Should occur automatically after selecting an attacking group (200.4) | The prompt should appear after selecting the attacking group. |  |
| 201.2 | Check if GamePlay records the selected group as the defending group.   1. Select a group 2. Check the group stored by GamePlay 3. Repeat steps 1 and 2 for multiple groups | The group recorded by GamePlay matches the selected group in every test. |  |
| 201.3 | Check if 4 is added to the attack modifier for each identical alignment   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with one identical alignment and no opposite alignments 3. Run code to calculate the attack modifier for identical alignments 4. Repeat steps 1-3 for groups with zero identical alignments and two or more identical alignments | The attack modifier should be equal to (4 \* Number of identical alignments) in every test. |  |
| 201.4 | Check if 4 is subtracted from the attack modifier for each opposite alignment   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with one opposite alignment and no identical alignments 3. Run code to calculate the attack modifier for opposite alignments 4. Repeat steps 1-3 for groups with zero opposite alignments and two or more opposite alignments | The attack modifier should be equal to (-4 \* Number of opposite alignments) in every test. |  |
| 201.5 | Check if (attacker power - defender resistance) is added to attack modifier   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with a known power and resistance 3. Run code to add (attacker power - defender resistance) to the attack modifier 4. Repeat steps 1-3 for groups with varying power and resistance (include cases where power > resistance, power = resistance, power < resistance). | The attack modifier should be equal to (attacker power - defender resistance) in every test. |  |
| 201.2.1 | Check if an error prompt occurs when the attacking group is selected as the defending group   1. Set the attacking group in GamePlay 2. Select the same group as the defending group as in 201.2 | A prompt should appear stating the defending group is invalid because it cannot attack itself. |  |
| 201.2.2 | Check if an error prompt occurs when the defending group is already owned by the attacking player   1. Select a group the attacking player already owns in 201.2 | A prompt should appear stating the defending group is invalid because it is already controlled by the player. |  |
| 201.2.3 | Check if an error prompt occurs when the defending group is owned by *The Discordian Society* and the attacking group is Government or Straight   1. Set attacking group as a group that is Government or Straight 2. Select a defending group whose owner is *The Discordian Society* in 201.2 | A prompt should appear stating the defending group is invalid because *The Discordian Society* is immune to attacks from Government and Straight groups. |  |
| 201.5.1 | Check if the resistance of *Gun Lobby* is set correctly   1. Set attack modifier to 0 2. Disable attack modifier adjustments for alignments 3. Set attacking group as a group that is Communist, Liberal, or Weird 4. Select *Gun Lobby* as the defending group 5. Repeat steps 3 and 4 with an attacking group that is not Communist, Liberal, or Weird | The attack modifier should equal (attacker power - 10) if the attacking group is Communist, Liberal, or Weird. Otherwise, the attack modifier should equal (attacker power - 3) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 202.1 | Check if the attacking player is prompted for the group to attack.   1. Identical to 201.1 | The prompt should appear after selecting the attacking group. |  |
| 202.2 | Check if GamePlay records the selected group as the defending group.   1. Identical to 201.2 | The group recorded by GamePlay matches the selected group in every test. |  |
| 202.3 | Check if +6 is added to the attack modifier   1. After the defending group is selected, check if the attack modifier is 6 | The attack modifier should be equal to 6. |  |
| 202.4 | Check if 4 is added to the attack modifier for each identical alignment   1. Identical to 201.3 | The attack modifier should be equal to (4 \* Number of identical alignments) in every test. |  |
| 202.5 | Check if 4 is subtracted from the attack modifier for each opposite alignment   1. Identical to 201.4 | The attack modifier should be equal to (-4 \* Number of opposite alignments) in every test. |  |
| 202.6 | Check if (attacker power - defender resistance) is added to attack modifier   1. Identical to 201.5 | The attack modifier should be equal to (attacker power - defender resistance) in every test. |  |
| 202.2.1 | Check if an error prompt occurs when the attacking group is selected as the defending group   1. Identical to 201.2.1 | A prompt should appear stating the defending group is invalid because it cannot attack itself. |  |
| 202.2.2 | Check if an error prompt occurs when the defending group is owned by *The Discordian Society* and the attacking group is Government or Straight   1. Identical to 201.2.3 | A prompt should appear stating the defending group is invalid because *The Discordian Society* is immune to attacks from Government and Straight groups. |  |
| 202.3.1 | Check if +4 is added to the attack modifier if the attacking player is *The Society of Assassins*   1. Set the attacking player to *The Society of Assassins* 2. Run 202.3 | The attack modifier should be equal to 10. |  |
| 202.6.1 | Check if the resistance of *Gun Lobby* is set correctly   1. Identical to 201.5.1 | The attack modifier should equal (attacker power - 10) if the attacking group is Communist, Liberal, or Weird. Otherwise, the attack modifier equals (attacker power - 3) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 203.1 | Check if the attacking player is prompted for the group to attack.   1. Identical to 201.1 | The prompt should appear after selecting the attacking group. |  |
| 203.2 | Check if GamePlay records the selected group as the defending group.   1. Identical to 201.2 | The group recorded by GamePlay matches the selected group in every test. |  |
| 203.3 | Check if 4 is added to the attack modifier for each opposite alignment   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with one opposite alignment and no identical alignments 3. Run code to calculate the attack modifier for opposite alignments 4. Repeat steps 1-3 for groups with zero opposite alignments and two or more opposite alignments | The attack modifier should be equal to (4 \* Number of opposite alignments) in every test. |  |
| 203.4 | Check if 4 is subtracted from the attack modifier for each identical alignment   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with one identical alignment and no opposite alignments 3. Run code to calculate the attack modifier for identical alignments 4. Repeat steps 1-3 for groups with zero identical alignments and two or more identical alignments | The attack modifier should be equal to (-4 \* Number of identical alignments) in every test. |  |
| 203.5 | Check if (attacker power - defender power) is added to attack modifier   1. Set the attack modifier to 0 2. Set the attacking and defending groups in GamePlay to groups with a known powers 3. Run code to add (attacker power - defender power) to the attack modifier 4. Repeat steps 1-3 for groups with varying powers (include cases where attacker power > defender power, attacker power = defender power, attacker power < defender power). | The attack modifier should be equal to (attacker power - defender power) in every test. |  |
| 203.2.1 | Check if an error prompt occurs when the attacking group is selected as the defending group   1. Identical to 201.2.1 | A prompt should appear stating the defending group is invalid because it cannot attack itself. |  |
| 203.2.2 | Check if an error prompt occurs when the selected group has 0 power   1. Select a defending group that has 0 power in 203.2 | A prompt should appear stating that a group with 0 power cannot be destroyed |  |
| 203.2.3 | Check if the attack proceeds as normal if *Whispering Campaign* is played and the defending group has 0 power   1. Set *Whispering Campaign* flag as true 2. Select a defending group that has 0 power in 203.2 | The prompt in error test case 203.2.2 does not show. The attack proceeds as normal. |  |
| 202.2.4 | Check if an error prompt occurs when the defending group is owned by *The Discordian Society* and the attacking group is Government or Straight   1. Identical to 201.2.3 | A prompt should appear stating the defending group is invalid because *The Discordian Society* is immune to attacks from Government and Straight groups. |  |
| 203.5.1 | Check if 2 is added to the attack modifier if the attacking player is *The Servants of Cthulu*   1. Set the attacking player to *The Servants of Cthulu* 2. Follow 203.5 | The attack modifier should be equal to ((attacker power + 2) - defender power) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 204.1 | Check if the distance between the defending group and the Illuminati is calculated correctly   1. Create a power structure with groups at varying distances from the Illuminati (e.g. from 1-5 cards away) 2. Calculate the distance of each card 3. Check the results with the actual distance of each card. | The calculated distance should match the actual distance |  |
| 204.2 | Check if the attack modifier is adjusted correctly based on distance between the defending group and the Illuminati   1. Create a power structure with groups at varying distances from the Illuminati (e.g. from 1-5 cards away) 2. Set attack modifier to 0 3. Set a card 1 away from the Illuminati as the defending group 4. Adjust the attack modifier based on the defending group’s distance from the Illuminati 5. Repeat steps 2-4 for cards between a distance of 2 and 5. | The attack modifier should equal -10 if the defending group is 1 away from the Illuminati. It should equal -5 if the defending group is 2 away from the Illuminati. It should equal -2 if the defending group is 3 away from the Illuminati. It should equal 0 if the defending group is 4 or more away from the Illuminati. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 205.1 | Check if the “any attempt” abilities are being checked properly   1. Select a group with an “any attempt” ability 2. Verify the check for an “any attempt” ability works properly 3. Repeat steps 1 and 2 when selecting a group without an “any attempt” ability | The check should return true if the group has an “any attempt” ability, and false if it does not. |  |
| 205.2 | Check if the “any attempt” ability attack types are being checked properly   1. Set the attack type to control 2. Select a group with an “any attempt” ability for an attack to control 3. Verify the check for an “any attempt” ability attack type works properly 4. Repeat steps 2 and 3 when the group does not have an “any attempt” ability for control 5. Repeat steps 1-4 when the attack types are set to neutralize and destroy | The check should return true if the group has an “any attempt” ability that matches the attack type, and false if it does not. |  |
| 205.3 | Check if the “any attempt” ability applies to the defending group by name   1. Select a group with an “any attempt” ability for a specific defending group 2. Set the defending group as the group listed on the ability 3. Verify the check for an “any attempt” ability for a specific group works properly 4. Repeat steps 1-3, but set the defending group to be different from the one on the ability | The check should return true if the group has an “any attempt” ability that matches the name of the defending group, and false if it does not. |  |
| 205.4 | Check if the “any attempt” ability applies to the defending group by alignment   1. Select a group with an “any attempt” ability for a specific alignment 2. Set the defending group as a group with the alignment on the ability 3. Verify the check for an “any attempt” ability for a specific alignment works properly 4. Repeat steps 1-3, but set the defending group to be one that does not have the alignment on the ability | The check should return true if the group has an “any attempt” ability that matches an alignment of the defending group, and false if it does not. |  |
| 205.5 | Check if all of the groups owned by the attacking player are checked for an “any attempt” ability   1. Create a power structure with many groups. 2. Run the loop that checks the entire power structure for “any attempt” abilities. 3. Print the name of the card to a console as it is checked 4. Compare the list of names to the cards in the power structure | The list of group names should match the cards that are in the power structure. |  |
| 203.6 | Check if the numerical value of the “any attempt” ability is added correctly to the attack modifier   1. Set the attack modifier to 0 2. Select a group with an “any attempt” ability 3. Select a defending group that fulfills the criteria of the “any attempt” ability 4. Calculate the attack modifier 5. Repeat steps 1-4 with varying numerical amounts of the “any attempt” ability | The attack modifier equals the numerical value of the “any attempt” ability |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 206.1 | Check if the “direct control” abilities are being checked properly   1. Select a group with an “direct control” ability 2. Verify the check for a “direct control” ability works properly 3. Repeat steps 1 and 2 when selecting a group without a “direct control” ability | The check should return true if the group has a “direct control” ability, and false if it does not. |  |
| 206.2 | Check if the attack type is checked for control correctly   1. Set the attack type to control 2. Verify the check returns true 3. Repeat steps 1 and 2 when setting the attack type to neutralize and control | The check should return true when the attack type is control, and false otherwise. |  |
| 206.3 | Check if the “direct control” ability applies to the defending group by name   1. Select a group with an “direct control” ability for a specific defending group 2. Set the defending group as the group listed on the ability 3. Verify the check for an “direct control” ability for a specific group works properly 4. Repeat steps 1-3, but set the defending group to be different from the one on the ability | The check should return true if the group has an “direct control” ability that matches the name of the defending group, and false if it does not. |  |
| 206.4 | Check if the “direct control” ability applies to the defending group by alignment   1. Select a group with an “direct control” ability for a specific alignment 2. Set the defending group as a group with the alignment on the ability 3. Verify the check for an “direct control” ability for a specific alignment works properly 4. Repeat steps 1-3, but set the defending group to be one that does not have the alignment on the ability | The check should return true if the group has an “direct control” ability that matches an alignment of the defending group, and false if it does not. |  |
| 206.5 | Check if the numerical value of the “direct control” ability is added correctly to the attack modifier   1. Set the attack modifier to 0 2. Select a group with an “direct control” ability 3. Select a defending group that fulfills the criteria of the “direct control” ability 4. Calculate the attack modifier 5. Repeat steps 1-4 with varying numerical amounts of the “direct control” ability | The attack modifier equals the numerical value of the “direct control” ability |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 207.1 | Check if the attack type is checked for control correctly   1. Identical to 206.2 | The check should return true when the attack type is control, and false otherwise. |  |
| 207.2 | Check if the defending card is checked for the government alignment correctly   1. Set the defending group to a group that has the government alignment 2. Verify the check returns true 3. Repeat steps 1 and 2 when setting the defending group to a group that does not have the government alignment | The check should return true when the defending group has the government alignment, and false otherwise. |  |
| 207.3 | Check that the attack modifier is adjusted correctly using *Chinese Campaign Donors*’sability   1. Set attack modifier to 0 2. Set the attacking group to *Chinese Campaign Donors*. 3. Set the defending group to a group with the government alignment. 4. Calculate the attack modifier based on the *Chinese Campaign Donors* ability 5. Repeat steps 1-3 with setting a defending group that does not have the government alignment. 6. Repeat steps 1-4 with setting an attacking group that is not *Chinese Campaign Donors* | The attack modifier should equal 4 if the attacking group is *Chinese Campaign Donors* and the defending group has the government alignment. The attack modifier should equal 0 for all of the other tests. |  |
| 207.3.1 | Check that if the defending group is owned by *The Discordian Society*, the player is prompted that the defending group is immune, and is prompted to select another defending group.   1. Set the attacking group to *Chinese Campaign Donors* 2. Set the defending group as a group with the government alignment 3. Set the defending group’s owner to *The Discordian Society* 4. Check if the error prompt occurs | The error prompt should state that *The Discordian Society* is immune to government and straight groups. The player is then prompted to select another defending group. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 208.1 | Check if the *Survivalists* card is checked for properly   1. Select the *Survivalists* group 2. Verify the check for the *Survivalists* card works properly 3. Repeat steps 1 and 2 when selecting a group that is not *Survivalists* | The check should return true if the group is *Survivalists*, and false if it does not. |  |
| 208.2 | Check if all of the groups owned by the defending player are checked if it is *Survivalists*   1. Create a power structure with many groups. 2. Run the loop that checks the entire power structure for *Survivalists* 3. Print the name of the card to a console as it is checked 4. Compare the list of names to the cards in the power structure | The list of group names should match the cards that are in the power structure. |  |
| 208.3 | Check that if the defending player has *Survivalists*, reduce the attack modifier by 2.   1. Set the attack modifier to 0 2. Add *Survivalists* to the defending player’s power structure 3. Run the check for *Survivalists* (208.2) 4. Repeat steps 1 and 3, ensuring the defending player does not own *Survivalists* | The attack modifier equals 2 if the defending player owns *Survivalists*. The attack modifier equals 0 otherwise. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 209.1 | Check that when the attacking player selects a group to aid an attack, that the group is added to the list of aiding groups and the transferable power is added to attack modifier.   1. Create a power structure 2. Set the attack modifier to 0 3. Set the list of aiding groups as an empty array 4. Add a group to aid the attack 5. Repeat steps 1-3 for varying numbers of groups, including 0 | The list of aiding groups should be all of the groups selected by the attacking player. The attack modifier should be equal to the sum of all of the aiding groups’ transferable power. |  |
| 209.2 | Check that when the attacking player selects a group to stop aiding an attack, that the group is removed from the list of aiding groups, and the transferable power is added to attack modifier   1. Create a power structure 2. Set the attack modifier to 0 3. Add several groups to aid in the attack 4. Remove a group from aiding in the attack 5. Repeat steps 1-3 for varying numbers of groups | The group should be removed from the list of aiding groups when removed from the attack. The attack modifier should be equal to the sum of all of the remaining aiding groups’ transferable power. |  |
| 209.1.1 | Check if an error prompt occurs if an aiding group’s attack counter is zero (meaning it has already attacked as many times as it could this turn)   1. Set a group’s attack counter to zero 2. Select that group to aid in the attack | A prompt should appear stating the aiding group is invalid because it has already used its attacks this turn. |  |
| 209.1.2 | Check if an error prompt occurs if the selected group has no transferable power   1. Add a group with no transferable power to the power structure 2. Select that group to aid in the attack | A prompt should appear stating the aiding group is invalid because it has no transferable power. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 210.1 | Check that this ability is only usable if the player is *The Bavarian Illuminati*, has not used the ability this turn, and has at least 5 MB in the Illuminati treasury   1. Set the player as *The Bavarian Illuminati* 2. Set playerAbilityUsed as False. 3. Set Illuminati treasury to 5 MB 4. Verify that this check runs properly 5. Repeat steps 1-4 for a different Illuminati 6. Repeat steps 1-5 for playerAbilityUsed = True 7. Repeat steps 1-6 for 0 MB in Illuminati treasury | The check should return true if the player is *The Bavarian Illuminati*, playerAbilityUsed as False, and there is at least 5 MB in the Illuminati treasury. It returns false otherwise. |  |
| 210.2 | Check that if the player is prompted if he or she wants to use the special ability   1. Set the player as *The Bavarian Illuminati* and playerAbilityUsed as False 2. Run this function, see if the prompt appears | The player should be prompted if he or she wants to use the special ability |  |
| 210.3 | Check the player’s response is recorded properly   1. Click on “yes” 2. Print the player’s selection 3. Repeat step 1, but click on “no” | Check that the player’s selection is correctly printed |  |
| 210.4 | Check that the attack is correctly set to privileged   1. Select “yes” in 210.3 2. Check that privilegeStatus is set to “privileged” in GamePlay 3. Repeat steps 1 and 2, but select “no” in 210.3 | The attack is set to privileged is “yes” is selected, and is not set to privileged if “no” is selected. |  |
| 210.5 | Check that 5 MB is removed from the Illuminati treasury   1. Set the Illuminati treasury to 5 MB 2. Run the script to remove 5 MB from the Illuminati treasury | The Illuminati treasury should contain 0 MB |  |
| 210.6 | Check that 5 MB is added to the bank   1. Set the bank to 0 MB 2. Run the script to add 5 MB to the bank | The bank should contain 5 MB |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 211.1 | Check that the player is prompted to discard a special to make an attack privileged only if the attacking player has at least one special card and the attack has never been privileged   1. Add a special card to the attacking player’s hand 2. Set privilegeStatus in GamePlay to “Never” 3. Run the script to see if the prompt works 4. Repeat steps 1-3 without adding a special card and instead removing all special cards from the attacking player’s hand 5. Repeat steps 1-4, instead setting privilegeStatus to “Privileged” or “Abolished” | The player should be prompted to make an attack privileged only if he or she holds at least one special card and privilegeStatus is “Never”. Otherwise, the prompt should not occur |  |
| 211.2 | Check the player’s response is recorded properly   1. Identical to 210.3 | Check that the player’s selection is correctly printed |  |
| 211.3 | Check that the player is prompted to select which card to discard   1. Select “yes” in 211.2 | The player is prompted to select a card to discard |  |
| 211.4 | Check that the card the player selected is discarded properly   1. Select a special card to discard 2. Check that the card is in the discard pile 3. Check that the special card is no longer owned by the player 4. Repeat steps 1-3 with 1 special card owned, and 2+ special cards owned | The selected card is no longer owned by the player and is in the discard pile. No other special cards are affected. |  |
| 211.5 | Check that the attack is set to privileged   1. Discard a card in 211.4 2. Check privilegeStatus in GamePlay | privilegeStatus should be set to “Privileged” |  |
| 211.4.1 | Check that if the player selects cancel, no special is discarded and the attack is not privileged   1. Set the player’s hand to one special card. 2. Select “cancel” 3. Check the player’s hand 4. Check privilegeStatus in GamePlay | The player’s hand should still have the one special card, and privilegeStatus should be “None” |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 212.1 | Check that each player is properly checked for the *Deep Agent* card   1. Add *Deep Agent* to one opposing player’s hand 2. Run the script checking for *Deep Agent* 3. Repeat steps 1 and 2 for each player (including the attacking player) 4. Repeat steps 1 and 2 without adding *Deep Agent* to anyone’s hand | Return true if a player other than the attacking player holds *Deep Agent*, false otherwise |  |
| 212.2 | Check that an opposing player is prompted to use *Deep Agent* if he or she has it   1. Add *Deep Agent* to your hand 2. Set another player as the attacking player 3. Run 212.1 | Prompts the player if he or she wants to use the *Deep Agent* card to abolish privilege |  |
| 212.3 | Check the player’s response is recorded properly   1. Identical to 210.3 | Check that the player’s selection is correctly printed |  |
| 212.4 | Check that *Deep Agent* is discarded correctly   1. Add *Deep Agent* under the player’s control 2. Run the script to discard it 3. Check that *Deep Agent* is in the discard pile 4. Check that *Deep Agent* is no longer owned by the player 5. Repeat steps 1-3 with 1 special card owned, and 2+ special cards owned | The *Deep Agent* is no longer owned by the player and is in the discard pile. No other special cards are affected. |  |
| 212.5 | Check that privilegeStatus is set correctly   1. After 212.3, print the value of privilegeStatus | If the player’s response is “yes”, privilegeStatus should be “Abolished”. If the player’s response is no, privilegeStatus should be “Privileged”. |  |
| 212.6 | Check that *Secrets Man Was Not Meant to Know* cancels the effect of *Deep Agent*   1. Add *Deep Agent* to one player’s hand 2. Add *Secrets Man Was Not Meant to Know* to another player’s hand 3. Use *Deep Agent*, then *Secrets Man Was Not Meant to Know* | *Deep Agent* and *Secrets Man Was Not Meant to Know* are both in the discard pile, and are no longer in the players’ hands. privilegeStatus should be “Privileged”. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 213.1 | Check that the player is prompted to discard two specials to abolish privilege only if the opposing player has at least two special cards and the attack is privileged   1. Add two special cards to the opposing player’s hand 2. Set privilegeStatus in GamePlay to “Privileged” 3. Run the script to see if the prompt works 4. Repeat steps 1-3 instead adding zero or one special cards 5. Repeat steps 1-4, instead setting privilegeStatus to “None” or “Abolished” | The player should be prompted to abolish privilege only if he or she has at least two special cards and the privilegeStatus is “Privileged”. Otherwise the prompt shouldn’t occur |  |
| 213.2 | Check the player’s response is recorded properly   1. Identical to 210.3 | Check that the player’s selection is correctly printed |  |
| 213.3 | Check that if a player selects a card to discard, the special card is added to discardList, and the discardCounter is increased by 1   1. Add many special cards to player’s hand 2. Select one card to discard 3. Print the values of discardList and discardCounter 4. Repeat steps 1 and 2 with 2 cards | The cards in discardList match the cards selected, and discardCounter matches the number of cards selected. |  |
| 213.4 | Check that the selected specials are discarded properly   1. Select two special cards to discard 2. Check that the cards are in the discard pile 3. Check that the special cards are no longer owned by the player 4. Repeat steps 1-3 with 2 special cards owned, 3+ special cards owned 5. Repeat steps 1-3 selecting a number of cards to discard other than 2 | The selected cards are no longer owned by the player and are in the discard pile. No other special cards are affected. This only occurs if the discardCounter is 2. |  |
| 213.5 | Check that privilegeStatus is set correctly   1. Set player response in 213.2 to “yes” 2. Set discardCounter to 2 3. Repeat step 1 setting player response to “no” 4. Repeat step 1-2, setting discardCounter to 1 | privilegeStatus should be “Abolished” after steps 1 and 2. privilegeStatus should be “Privileged” after steps 3 and 4. |  |
| 213.3.1 | Check that if the player selects cancel, no specials are discarded and the privilege is not abolished   1. Set the player’s hand to two special cards. 2. Select “cancel” 3. Check the player’s hand 4. Check privilegeStatus in GamePlay | The player’s hand should still have the two special cards, and privilegeStatus should be “Privileged” |  |
| 213.3.2 | Check that if player clicks on a card that is already selected to be discarded, it is removed from discardList, and discardCounter is decreased by 1.   1. Add many special cards to player’s hand 2. Select one card to discard 3. Select the card again to cancel that choice 4. Print the values of discardList and discardCounter 5. Repeat steps 1 and 2 with 2 cards | The cards in discardList match the cards selected, and discardCounter matches the number of cards selected. |  |
| 213.4.1 | Check that *Secrets Man Was Not Meant to Know* cancels the effect of one of the special cards   1. Add *Secrets Man Was Not Meant to Know* to an opposing player’s hand 2. Repeat 213.4 3. Play *Secrets Man Was Not Meant to Know* 4. Print contents of discardList and discardCounter | *Secrets Man Was Not Meant to Know* is in the discard pile and not in the opposing player’s hand. discardList contains the two specials selected to be discarded, discardCounter is equal to 1. Abolishing player is prompted to discard 1 more special card. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 214.1 | Check that the “Cancel Attack” button reacts to the player properly   1. Output a message to the console at the beginning of the function called by pressing “Cancel Attack”. 2. Set moneySpentOnAttack in GamePlay to false 3. Set diceRoll in GamePlay = 0 4. Press “Cancel Attack” 5. Repeat steps 2-4 with different values for moneySpentOnAttack and diceRoll | The console should show the message placed in the function only if moneySpentOnAttack is false and diceRoll is 0. Otherwise the “Cancel Attack” button should be disabled. |  |
| 214.2 | Check that all players are prompted that the attack has been canceled.   1. Run the script 2. Check for the prompt on the attacking player, defending player, and other players | All players should receive the prompt that the attack has been canceled. |  |
| 214.3 | Check that the attacking player returns to action selection, and the player has not lost an action   1. Set actionsRemaining to 2 2. Press “Cancel Attack” 3. Check that the action menu has returned 4. Check actionsRemaining | The action menu has returned, and actionsRemaining is equal to 2. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 215.1 | Check if the attacking player is prompted to spend money on the attack   1. Run the code for the prompt | The prompt should appear prompting the player to spend money on the attack. |  |
| 215.2 | Check the attack modifier is adjusted properly based on the amount of MB spent from the Illuminati treasury. Check that the MB is transferred properly from the Illuminati treasury to the bank   1. Set the attack modifier to 0 2. Set MB of Illuminati treasury to 10 3. Set MB of bank to 0 4. Follow the prompt on 215.1 to spend money 5. Check the attack modifier, Illuminati treasury, and bank 6. Repeat steps 1-5 for varying amounts of MB spent | The attack modifier is equal to the amount of MB spent. The Illuminati treasury should have (10 - amount spent) MB. The bank should have MB equal to the amount spent. |  |
| 215.3 | Check the attack modifier is adjusted properly based on the amount of MB spent from the attacking group’s treasury. Check that the MB is transferred properly from the attacking group’s treasury to the bank   1. Set the attack modifier to 0 2. Set a group as the attacking group 3. Set MB of attacking group’s treasury to 10 4. Set MB of bank to 0 5. Follow the prompt on 215.1 to spend money 6. Check the attack modifier,attacking group’s treasury, and bank 7. Repeat steps 1-6 for varying amounts of MB spent | The attack modifier is equal to the amount of MB spent. The attacking group’s treasury should have (10 - amount spent) MB. The bank should have MB equal to the amount spent. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 216.1 | Check if the defending player is prompted to spend money on the attack   1. Run the code for the prompt | The prompt should appear prompting the player to spend money on the attack. |  |
| 216.2 | Check the attack modifier is adjusted properly based on the amount of MB spent from the Illuminati treasury. Check that the MB is transferred properly from the Illuminati treasury to the bank   1. Set the attack modifier to 0 2. Set MB of Illuminati treasury to 10 3. Set MB of bank to 0 4. Follow the prompt on 216.1 to spend money 5. Check the attack modifier, Illuminati treasury, and bank 6. Repeat steps 1-5 for varying amounts of MB spent | The attack modifier is equal to the negative of the amount of MB spent. The Illuminati treasury should have (10 - amount spent) MB. The bank should have MB equal to the amount spent. |  |
| 216.3 | Check the attack modifier is adjusted properly based on the amount of MB spent from the defending group’s treasury. Check that the MB is transferred properly from the attacking group’s treasury to the bank   1. Set the attack modifier to 0 2. Set a group as the defending group 3. Set MB of defending group’s treasury to 10 4. Set MB of bank to 0 5. Follow the prompt on 216.1 to spend money 6. Check the attack modifier,defending group’s treasury, and bank 7. Repeat steps 1-6 for varying amounts of MB spent | The attack modifier is equal to the negative of the amount of MB spent multiplied by 2. The attacking group’s treasury should have (10 - amount spent) MB. The bank should have MB equal to the amount spent. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 217.1 | Check if privilege is accounted for correctly   1. Print to console if the attack is privileged | The console should display true if privilegeStatus is “Privileged”, and false if privilegeStatus is “None” or “Abolished” |  |
| 217.2 | Check if the defending player is prompted to spend money on the attack   1. Run the code for the prompt | The prompt should appear prompting the player to spend money on the attack. |  |
| 217.3 | Check the attack modifier is adjusted properly based on the amount of MB spent from the Illuminati treasury. Check that the MB is transferred properly from the Illuminati treasury to the bank   1. Set the attack modifier to 0 2. Set MB of Illuminati treasury to 10 3. Set MB of bank to 0 4. Follow the prompt on 217.2 to spend money 5. Select to aid the defender 6. Check the attack modifier, Illuminati treasury, and bank 7. Repeat steps 1-6 for varying amounts of MB spent 8. Repeat steps 1-6 but instead aid the attacker | The attack modifier is equal to the of the amount of MB spent if aiding the attacker, and the negative of the amount of MB spent if aiding the defender. The Illuminati treasury should have (10 - amount spent) MB. The bank should have MB equal to the amount spent. |  |
| 217.4 | Check that all players that are not the attacking or defending player are allowed to interfere   1. Run the script, ensuring that all players receive the prompt | All players should receive the prompt and have the opportunity to spend money to interfere. |  |
| 217.1.1 | Check that if the attack is privileged, and a player plays *Interference* he or she is allowed to interfere   1. Set privilegeStatus to “Privileged” 2. Give an opposing player *Interference* 3. Play *Interference* 4. Check if the player is prompted to interfere | The player(s) who play interference is prompted and given the opportunity to interfere in the attack. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 218.1 | Check that the spending phase only ends when no money is spent by any player.   1. Set moneySpentThisPhase in GamePlay to 0 2. Run the check at the end of the spending phase 3. Repeat step 1 instead using a number >0 | The spending phase should end only if moneySpentThisPhase is equal to 0. Otherwise the spending phase begins again |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 219.1 | Check if the die is generating random numbers from 2 to 12, inclusively   1. Run function Die.roll() to get a large sample size of values (e.g. 100) | Check if all of the results are all integers between 2 and 12, inclusively |  |
| 219.2 | Check if a roll less than the attack modifier and less than 11 is a success   1. Instead of using Die.roll(), give a specific input for the roll and attack modifier | A roll less than the attack modifier and less than 11 should return a success   1. For example, set roll to 7 and attack modifier to 8, check that the attack is a success. 2. Set roll to 11 and attack modifier to 8, check that attack is a failure |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 220.1 | Check if the attacking player is prompted to place the newly acquired group   1. Set attackSuccessful in GamePlay to true 2. Run the script | Player should be prompted to place the newly acquired group into the power structure |  |
| 220.2 | Check that the selected outward control arrow is connected to the new group   1. Create a power structure 2. Set a group as the attacking group (parent group) 3. Set a group as the defending group (new group) 4. Select an open control arrow where the new group will fit. 5. Check the parent group’s child at that control arrow and the new group’s parent | The parent group’s child at the selected control arrow should be the new group, and the new group’s parent should be the parent group. |  |
| 220.3 | Check that the new group’s children are placed on the same control arrows as they were prior to the attack.   1. Create a power structure 2. Set a group as the attacking group (parent group) 3. Set a group as the defending group (new group) 4. Add children to the defending group 5. Select an open control arrow where the new group and all of its children will fit. 6. Check the new group’s children remain on the same control arrows. 7. Repeat steps 1-6, but add children to the children, and so forth. | The new group’s children should remain on the same control arrows as they were prior to the attack in the new power structure. This applies to children of the children, etc. |  |
| 220.4 | Check that the treasuries of the new group and all of its children are cut in half (rounded down), with the MB going to the bank.   1. Create a power structure 2. Set a group as the attacking group (parent group) 3. Set a group as the defending group (new group) 4. Add children to the defending group 5. Set the MB of the new group and all children to 10. 6. Set the MB in the bank to 0. 7. Select an open control arrow where the new group and all of its children will fit. 8. Check the MB of each group and the bank 9. Repeat steps 1-8 with various amounts of MB on the groups. | The new group and its children should have its MB reduced in half (rounded down). The amount in the bank should equal the amount of MB removed from all of the children. |  |
| 220.5 | Check if the attacking player is prompted to transfer MB to the new group   1. Complete 220.4 | Player should be prompted to transfer MB to the new group |  |
| 220.6 | Check that the MB is properly transferred from the parent group to the new group   1. Set the MB of the parent group to 10 2. Set the MB of the child group to 10 3. Set the amount of MB to transfer to 5 | The parent group should have 5 MB, and the child group should have 15 MB. |  |
| 220.7 | Check that the attack counter of the attacking group is reduced by 1   1. Set the attack counter of the attacking group to 1 2. Run the script 3. Check the attack counter 4. Repeat steps 1-3, instead setting the attack counter to 2 | The attack counter of the attacking group should equal 0 in the first test, and 1 in the second test. |  |
| 220.8 | Check that the attack counter of the aiding groups is all reduced by 1   1. Add groups to the list of aiding groups 2. Set the attack counter of each of these groups to 1 3. Run the script 4. Check the attack counter of each aiding group 5. Repeat steps 1-3, instead setting the attack counter to 2 | The attack counter of all of the aiding groups should equal 0 in the first test, and 1 in the second test. |  |
| 220.3.1 | Check that the player is prompted to move a child of the new group if it cannot fit   1. Create a power structure 2. Set a group as the attacking group (parent group) 3. Set a group as the defending group (new group) 4. Add children to the defending group 5. Select an open control arrow where the new group will fit, but not all of its children 6. Confirm that the player is prompted to move the child that will not fit | The player is prompted to move the child if it cannot fit in the power structure if it stays on the same control arrow. |  |
| 220.3.2 | Check that if a child of the new group must change control arrows (220.3.1), the selected outward control arrow is connected to the new group   1. Select a new control arrow for the child that cannot fit. Make sure the child will fit in this control arrow. 2. Check the parent group’s child at the selected control arrow and the new group’s parent. Check the child at the old control arrow. | The parent group’s child at the selected control arrow should be the moved child, and the moved child’s parent should be the parent group. The child at the old control arrow should be null. |  |
| 220.3.3 | Check that if the child cannot fit in the power structure and cannot fit on any of the parent’s control arrows, that the child and all of its children are made uncontrolled. Check that the MB of all of these groups are placed in the bank.   1. Set a group as the attacking group (parent group) 2. Set a group as the defending group (new group) 3. Add children to the defending group 4. Set the treasuries of the new group and all of its children to 10 MB 5. Set bank’s MB to 0 MB 6. Create a power structure so that the new group cannot fit on any of the attacking group’s control arrows 7. Select an open control arrow 8. Confirm that the child and its children are uncontrolled. 9. Confirm that the now uncontrolled groups have 0 MB, and the bank has the MB they lost. | The new group and its children are uncontrolled (have no parent). Each of these groups should have 0 MB, and the bank should have (10 \* (number of groups that were made uncontrolled)) MB. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 221.1 | Check that the MB from the neutralized group and its children are reduced to 0 MB, and the MB is placed in the bank   1. Set the neutralized group and its children 2. Set each group’s treasury to 10 MB 3. Set the bank to 0 MB 4. Run the script | Each of the groups should have 0 MB, and the bank should have (10 \* (number of groups that were made uncontrolled)) MB. |  |
| 221.2 | Check that the neutralized group and its children are made uncontrolled.   1. Create the defending player’s power structure 2. Set the neutralized group as a group in the power structure with children 3. Run the script | The neutralized group and its children are made uncontrolled (have no parent) |  |
| 221.3 | Check that the attack counter of the attacking group is reduced by 1   1. Identical to 220.7 | The attack counter of the attacking group should equal 0 in the first test, and 1 in the second test. |  |
| 221.4 | Check that the attack counter of the aiding groups is all reduced by 1   1. Identical to 220.8 | The attack counter of all of the aiding groups should equal 0 in the first test, and 1 in the second test. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 222.1 | Check that the MB from the destroyed group and its children are reduced to 0 MB, and the MB is placed in the bank   1. Identical to 221.1 | Each of the groups should have 0 MB, and the bank should have (10 \* (number of groups that were made uncontrolled)) MB. |  |
| 222.2 | Check that the destroyed group is placed in the “Dead pile   1. Create the defending player’s power structure 2. Set the destroyed group as a group in the power structure 3. Run the script 4. Print to the console the names of the cards in the “Dead” pile. | The destroyed group is in the “Dead” pile |  |
| 222.3 | Check that the children of the destroyed group is placed in the uncontrolled area   1. Create the defending player’s power structure 2. Set the destroyed group as a group in the power structure with children 3. Run the script | The destroyed group’s children are made uncontrolled (have no parent) |  |
| 222.4 | Check that the attack counter of the attacking group is reduced by 1   1. Identical to 220.7 | The attack counter of the attacking group should equal 0 in the first test, and 1 in the second test. |  |
| 222.5 | Check that the attack counter of the aiding groups is all reduced by 1   1. Identical to 220.8 | The attack counter of all of the aiding groups should equal 0 in the first test, and 1 in the second test. |  |
| 222.6 | Check that if the attacking player is *The Servants of Cthulu* that the destroyedGroups counter is increased by 1   1. Set the attacking player’s Illuminati to *The Servants of Cthulu* 2. Set destroyedGroups in GamePlay to 0 3. Run the script 4. Print the value of destroyedGroups 5. Repeat steps 1-4 with a different Illuminati | The value of destroyedGroups should be set to 1 only if the attacking group is *The Servants of Cthulu* |  |

Specials:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 300.1 | Check if player is prompt to play Bribery Card.   1. Allows player to select uncontrolled group to bribe | Player will be prompt to select what special card to play from special cards in hand. After player selects to play bribery card, player will be prompt to choose what uncontrolled group to bribe |  |
| 300.2 | Checks if player’s choice of uncontrolled group to bribe is allowed.   1. Check uncontrolled groups in play | Player will be allowed to select uncontrolled group to bribe. |  |
| 300.3 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 300.4 | Checks if player’s choice of uncontrolled group to bribe is in player’s power structure. | Uncontrolled group player choices to bribe will be shown in player’s power structure. Player will get to use card in future plays |  |
| 300.5 | Checks if Bribery Card is used. | If player chooses to play Bribery Card, bribery card should be removed from player’s special cards after Test ID: 300.4 |  |
| 300.6 | Checks if player has at least one action remaining if player chooses to play Bribery Card.   1. Allows player to play card | Player will be allowed to play Bribery Card. |  |
| 300.7 | Checks if player’s actions are reduced by 1 after Bribery Card is played | Player will be allowed to choose another action if player has 1 action left. If player does not have 1 action left, the player’s turn ends. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 301.1 | Check if player is prompt to play Computer Espionage Card.   1. If player has Computer Espionage Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. After player selects to play Computer Espionage Card, player will prompt to either Count Money or Count Specials |  |
| 301.2 | Checks if player chooses to Count Money.   1. Allows player to select what group to Count Money. 2. Checks selected group to see if group card is in play. | Game will allow player to select Count Money. Immediately after player will be able to choose what group to Count Money from game. Game will only let player choose from selectable groups. |  |
| 301.3 | Checks if player chooses to Count Specials.   1. Allows player to select what opponent to Count Specials from. 2. Checks if opponent has special cards. 3. Opponents specials are displayed 4. Checks if opponent’s specials are displayed. | Player will be prompt to choose what opponent they want to see special cards. If opponent has special cards in hand, player will be allowed to see what special cards opponent has. Opponents specials are displayed to player. |  |
| 301.4 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 301.5 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 301.6 | Checks if Computer Espionage Card is used. | If player chooses to play Computer Espionage Card, Computer Espionage Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 302.1 | Check if player is prompt to play Media Campaign Card.   1. If player has Media Campaign Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. After player selects to play Media Campaign Card, player will prompt to pick group from “dead” pile. |  |
| 302.2 | Checks if “dead” pile has groups | Player will be prompt to select a group from “dead” pile to put in uncontrolled area. |  |
| 302.3 | Checks if player’s chosen card is put in uncontrolled area. | Card will be removed from “dead” pile and place in the uncontrolled area. |  |
| 302.4 | Checks if “dead” pile has no groups | Player will be prompt to pick a different special card. |  |
| 302.5 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 302.6 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 302.7 | Checks if Media Campaign Card is used. | If player chooses to play Media Campaign Card, Media Campaign Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 303.1 | Check if player is prompt to play Slush Fund Card.   1. If player has Slush Fund Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. Player selects to play Slush Fund Card. |  |
| 303.2 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 303.3 | Checks if 15 MB is added into player’s Illuminati treasury. | 15 MB is added into player’s illuminati treasury. |  |
| 303.4 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 303.5 | Checks if Slush Fund is used. | If player chooses to play Slush Fund Card, Slush Fund Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 304.1 | Check if player is prompt to play Swiss Bank Account Card.   1. If player has Swiss Bank Account Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. Player selects to play Swiss Bank Account Card. |  |
| 304.2 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 304.3 | Checks if 25 MB is added into player’s Illuminati treasury. | 25 MB is added into player’s illuminati treasury. |  |
| 304.4 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 304.5 | Checks if Swiss Bank Account is used. | If player chooses to play Swiss Bank Account Card, Swiss Bank Account Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 305.1 | Check if player is prompt to play Whispering Campaign Card.   1. If player has Whispering Campaign Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. After player selects to play Whispering Campaign Card, player will be allowed to choose group to attack. |  |
| 305.2 | Checks if player has at least one action remaining if player chooses to play Bribery Card.   1. Allows player to play card | Player will be allowed to play Whispering Campaign Card. |  |
| 305.3 | Checks if group player plans to attack, has no power   1. Group’s attack level is accessed | Player attacks group with no power. If group has power, player can not choose to attack group. |  |
| 305.4 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 305.5 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 305.6 | Checks if Whispering Campaign is used. | If player chooses to play Whispering Campaign Card, Whispering Campaign Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 306.1 | Check if player is prompt to play White Collar Crime Card.   1. If player has White Collar Crime Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. After player selects to play White Collar Crime Card, player will be allowed to choose between transferring money from group to group or transferring money from treasury. |  |
| 306.2 | Checks if player chooses to transfer group to group.   1. Player will be allowed to choose what group transfers to what other group. 2. Checks if group selected to transfer money has money. | Player is prompted to choose from controlled groups what group is going to transfer to another controlled group. After selection, receiving group gains money in their treasury and giving group loses money from their treasury. |  |
| 306.3 | Checks if giving group has money.   1. Access group’s money. | Group will be allowed to transfer money to another group, if group has money. Group loses money. |  |
| 306.4 | Checks if receiving group gains money. | Receiving group will gain money transferred. |  |
| 306.5 | Checks if player choose to transfer from treasury to group.   1. Player will be allowed to choose what group will receive money. | Player will be allowed to choose what receiving group will receive money from player’s illuminati treasury. |  |
| 306.6 | Checks if player’s treasury has money to give.   1. Accesses player’s illuminati treasury. | Player will be allowed to transfer money. When money is transferred, 5 MB will be subtracted from treasury. |  |
| 306.7 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 306.8 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 306.9 | Checks if White Collar Crime is used. | If player chooses to play White Collar Crime Card, White Collar Crime Card should be removed from player’s special cards. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Description** | **Expected Results** | **Actual Results** |
| 307.1 | Check if player is prompt to play Interference Card.   1. If player has Interference Card, player will be allowed to play card. | Player will be prompt to select what special card to play from special cards in hand. Player selects to play Interference Card |  |
| 307.2 | Checks if Privileged Attack is in play   1. Accesses Players’ recent played cards | If Privileged Attack is not in play, player will be prompt to select another special card to play. If privileged Attack is in play, card will be discard after use. |  |
| 307.3 | Checks if player selects to play “Secrets Man Was Not Meant to Know” card.   1. Checks if player has “Secrets Man Was Not Meant to Know” card. 2. Player will be prompt either not to play or play card, only if player has card. | Player will be allowed to select either to play or not to play “Secrets Man Was Not Meant to Know” card. |  |
| 307.4 | Checks if “Secrets Man Was Not Meant to Know” card is used. | If player chooses to play “Secrets Man Was Not Meant to Know”, card should be removed from player’s special cards. |  |
| 307.5 | Checks if Interference is used. | If player chooses to Interference Card, Interference Card should be removed from player’s special cards. |  |