|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| displayFloatingIcons | 1 | The spacing is less than 1 | nMax = 5  nSpacing = 0 | No characters will be printed | No characters printed | P |
|  | 2 | Max number of characters to be printed is less than 1 | nMax = 0  nSpacing = 5 | No characters will be printed | No characters printed | P |
|  | 3 | Spacing is less than the max number of characters to be printed. The icon to be printed will be printed at the indicated spacing from the start or last print. | nMax = 11  nSpacing = 7  cIcon = ‘!’ | To be printed:  -6 space characters  -1 icon character of choice (‘!’)  -4 space characters | Printed:  -6 space characters  -1 icon character of choice (‘!’)  -4 space characters | P |
|  | 4 | Spacing is greater than the ma max number of characters to be printed. The icon will not be printed. Only spaces will be printed | nMax = 10  nSpacing = 11 | To be printed:  -10 space characters | Printed:  -10 space characters | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| getSingleChar | 1 | User entered an invalid string input; empty string; newline character excluded | sScan = ‘’ | \*cChar1 = ‘’ | \*cChar1 = ‘’ | P |
|  | 2 | User entered a valid character input | sScan = ‘y’ | \*cChar1 = ‘y’ | \*cChar1 = ‘y’ | P |
|  | 3 | User entered more than 1 character | User input:  YnN <\n> | \*cChar1 = ‘Y’ | \*cChar1 = ‘Y’ | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| getString | 1 | User entered an invalid string input; empty string; newline character excluded | sString = “” /\* new line character is not included\*/ | 0 | 0 | P |
|  | 2 | User entered a cancel character for string input cancelation. | \*cCancelChar = “~”  sString = “~” | -1 | -1 | P |
|  | 3 | User entered a valid string input | sString = “Word” | 1 | 1 | P |
|  | 4 | Entered string exceeds the required string length | Entered string: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  nLength = 20 | 1  sString =  “ABCDEFGHIJKLMNOPQRST” | 1  sString =  “ABCDEFGHIJKLMNOPQRST” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| generateRandomNum | 1 | Generate random number between 0 and number of elements in the array of entries, including 0. | nMaxLim = 50 | Returns an int value that is 0 <= int value < 50 | 42 | P |
|  | 2 | Generate random number between 0 and number of clues in a single entry, including 0. | nMaxLim = 7 | Returns an int value that is 0 <= int value < 7 | 6 | P |
|  | 3 | Indicated max limit for the range of random numbers to be executed is less than or equal to 0. | nMaxLim = 0 | -1 | -1 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| delay | 1 | No delay | nSeconds = 0 | Program pauses for 0 seconds, then heads to the next line of instruction. | Paused for 0 seconds, then heads to the next line of instruction. | P |
|  | 2 | Delay for 3 seconds | nSeconds = 3 | Program pauses for 3 seconds, then heads to the next line of instruction. | Paused for 3 seconds, then heads to the next line of instruction. | P |
|  | 3 | Delay for 5 seconds | nSeconds = 5 | Program pauses for 5 seconds, then heads to the next line of instruction. | Paused for 5 seconds, then heads to the next line of instruction. | P |
|  | 4 | Negative value of time | nSeconds = -2 | Skips the function and heads to the next line of instruction | Skipped the function, then heads to the next line of instruction. | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| getWordEntry | 1 | User entered an invalid string input; empty string; newline character excluded | sChosenWord = “” | \*bValid = 0  Displays invalid input note and asks for string input again | \*bValid = 0  Displayed invalid input note | P |
|  | 2 | User entered a cancel character for string input cancelation. | Cancel Character: “~”  sChosenWord = “~” | \*bValid = -1  Exits the function | \*bValid = -1  Exited the function | P |
|  | 3 | User entered a valid string input | sChosenWord = “Loop” | \*bValid = 1  String in sChosenWord is kept, and exits the funciton | \*bValid = 1  String in sChosenWord kept, and exited the function. | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| IsUniqueLetter | 1 | The randomly chosen entry is not yet used in any row | nRandNum = 0  aEntry[0].use = 0 | 1 | 1 | P |
|  | 2 | The randomly chosen entry is used already. It can be present in any row. | nRandNum = 2  aEntry[2].use = 2  //current row index is 0  //gameboard size 2 row, 3 col | 0 | 0 | P |
|  | 3 | The randomely chosen entry’s first letter already exists in the same row | nRandNum = 0  aEntry[0].answer[0] = ‘A’  aEntry[0].use = 1  gameboardRow[0] = ‘A’  //current row index is 0  //gameboard size 2 row, 3 col | 0 | 0 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| checkLettersLeft | 1 | The entries that are not yet used for the gameboard has a set of unique first letters | nElem = 7  aEntries[0].answer = “Leaves”  aEntries[0].use = 0  aEntries[1].answer = “Ant”  aEntries[1].use = 1  aEntries[2].answer = “Apple Pie”  aEntries[2].use = 0  aEntries[3].answer = “Car”  aEntries[3].use = 0  aEntries[4].answer = “Zebra”  aEntries[4].use = 0  aEntries[5].answer = “Yoyo”  aEntries[5].use = 0  aEntries[6].answer = “Tea”  aEntries[6].use = 0  From File: (INIT1.txt)  Object: Leaves  Color: Green  Grows on: Trees  Object: Ant  Kind of: Insect  Object: Apple Pie  Kind of: Dessert  Object: Car  Size: Big  Object: Zebra  Color: Black and white  Object: Yoyo  Used for: Playing  Object: Tea  Helps in: Relaxation | 6 | 6 | P |
|  | 2 | The entries that are not yet used for the gameboard has a set of unique first letters | /\* 2 rows and 4 columns\*/  nElem = 7  aEntries[0].answer = “Ax”  aEntries[0].use = 1  aEntries[1].answer = “Ant”  aEntries[1].use = 0  aEntries[2].answer = “Apple Pie”  aEntries[2].use =  aEntries[3].answer = “Car”  aEntries[3].use = 1  aEntries[4].answer = “Ache”  aEntries[4].use = 0  aEntries[5].answer = “Age”  aEntries[5].use = 0  aEntries[6].answer = “Tea”  aEntries[6].use = 0  From File: (INIT2.txt)  Object: Ax  Usage: Wood cutting  Object: Ant  Kind of: Insect  Object: Apple Pie  Kind of: Dessert  Object: Car  Size: Big  Object: Ache  Synonym: Pain  Object: Age  Can be: Counted  Object: Tea  Helps in: Relaxation | 2 | 2 | P |
|  | 3 | All the entries are used already for the gameboard | nElem = 8  aEntries[0].answer = “Leaves”  aEntries[0].use = 1  aEntries[1].answer = “Ant”  aEntries[1].use = 1  aEntries[2].answer = “Apple Pie”  aEntries[2].use = 2  aEntries[3].answer = “Car”  aEntries[3].use = 1  aEntries[4].answer = “Zebra”  aEntries[4].use = 1  aEntries[5].answer = “Yoyo”  aEntries[5].use = 2  aEntries[6]answer = “Tea”  aEntries[6].use = 2  aEntries[7].answer = “Wheel”  aEntries[7].use = 2  From File: (INIT1.txt)  Object: Leaves  Color: Green  Grows on: Trees  Object: Ant  Kind of: Insect  Object: Apple Pie  Kind of: Dessert  Object: Car  Size: Big  Object: Zebra  Color: Black and white  Object: Yoyo  Used for: Playing  Object: Tea  Helps in: Relaxation | 0 | 0 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| checkGameboard | 1 | User letter input is not in the current row of gameboard | //currently in the 1st row  nCurrentRow = 1  nCol = 4  gameboardRow[4] = {‘A’,’W’,’S’,’D’}  cChoice = ‘T’  nElem = 5 | \*nIndex = -1  \*nCurrCol = -1 | \*nIndex = -1  \*nCurrCol = -1 | P |
|  | 2 | User letter input is in the current row of gameboard | //currently in the 1st row  nCurrentRow = 1  nCol = 4  gameboardRow[4] = {‘A’,’W’,’S’,’D’}  cChoice = ‘A’  nElem = 5  aEntries[0].answer[0] = ‘A’  aEntries[0].use = 1 | \*nIndex = 0  \*nCurrCol = 0 | \*nIndex = 0  \*nCurrCol = 0 | P |
|  | 3 | Player chose a specific letter that is also present in the other row | //currently in the 1st row  nCurrentRow = 1  nCol = 4  gameboardRow[4] = {‘A’,’W’,’S’,’D’}  cChoice = ‘A’  nElem = 5  aEntries[0].answer = “Ant”  aEntries[0].use = 2  aEntries[1].answer = “Wheel”  aEntries[2].answer = “Snake”  aEntries[3].answer = “Dog”  aEntries[4].answer = “Apex”  aEntries[4].use = 1 | \*nIndex = 4  \*nCurrCol = 0 | \*nIndex = 4  \*nCurrCol = 0 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| initializeEntries | 1 | Sets the strings of aEntries to empty strings, and the integers to zero | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[0].use = 3  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[1].numClues = 1  aEntries[1].use = 2  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting”  aEntries[2].numClues = 1  aEntries[2].use = 1 | aEntries[0].answer = “”  aEntries[0].clueList[0].relation = “”  aEntries[0].clueList[0].relValue = “”  aEntries[0].numClues = 0  aEntries[0].use = 0  aEntries[1].answer = “”  aEntries[1].clueList[0].relation = “”  aEntries[1].clueList[0].relValue = “”  aEntries[1].numClues = 0  aEntries[1].use = 0  aEntries[2].answer = “”  aEntries[2].clueList[0].relation = “”  aEntries[2].clueList[0].relValue = “”  aEntries[2].numClues = 0  aEntries[2].use = 0 | aEntries[0].answer = “”  aEntries[0].clueList[0].relation = “”  aEntries[0].clueList[0].relValue = “”  aEntries[0].numClues = 0  aEntries[0].use = 0  aEntries[1].answer = “”  aEntries[1].clueList[0].relation = “”  aEntries[1].clueList[0].relValue = “”  aEntries[1].numClues = 0  aEntries[1].use = 0  aEntries[2].answer = “”  aEntries[2].clueList[0].relation = “”  aEntries[2].clueList[0].relValue = “”  aEntries[2].numClues = 0  aEntries[2].use = 0 | P |
|  | 2 | The strings of entries are already set to empty strings and its integers to zero | aEntries[0].answer = “”  aEntries[0].clueList[0].relation = “”  aEntries[0].clueList[0].relValue = “”  aEntries[0].numClues = 0  aEntries[0].use = 0  aEntries[1].answer = “”  aEntries[1].clueList[0].relation = “”  aEntries[1].clueList[0].relValue = “”  aEntries[1].numClues = 0  aEntries[1].use = 0  aEntries[2].answer = “”  aEntries[2].clueList[0].relation = “”  aEntries[2].clueList[0].relValue = “”  aEntries[2].numClues = 0  aEntries[2].use = 0 | aEntries[0].answer = “”  aEntries[0].clueList[0].relation = “”  aEntries[0].clueList[0].relValue = “”  aEntries[0].numClues = 0  aEntries[0].use = 0  aEntries[1].answer = “”  aEntries[1].clueList[0].relation = “”  aEntries[1].clueList[0].relValue = “”  aEntries[1].numClues = 0  aEntries[1].use = 0  aEntries[2].answer = “”  aEntries[2].clueList[0].relation = “”  aEntries[2].clueList[0].relValue = “”  aEntries[2].numClues = 0  aEntries[2].use = 0 | aEntries[0].answer = “”  aEntries[0].clueList[0].relation = “”  aEntries[0].clueList[0].relValue = “”  aEntries[0].numClues = 0  aEntries[0].use = 0  aEntries[1].answer = “”  aEntries[1].clueList[0].relation = “”  aEntries[1].clueList[0].relValue = “”  aEntries[1].numClues = 0  aEntries[1].use = 0  aEntries[2].answer = “”  aEntries[2].clueList[0].relation = “”  aEntries[2].clueList[0].relValue = “”  aEntries[2].numClues = 0  aEntries[2].use = 0 | P |
|  | 3 | N/A | N/A | N/A | N/A | N/A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| addTriviaMenu | 1 | User enters an invalid string; empty string | strWordToAddTrivia = “” | Prints an invalid input note, and asks user for another string input | Printed invalid input note and asked for another string input | P |
|  | 2 | User cancels and exits in the Add Trivia option | Cancel character: “~”  strWordToAddTrivia = “~” | Exits the function | Exited the function | P |
|  | 3 | User entered a valid string input but string input doesn’t exist in the current entry list | strWordToAddTrivia = “Holo” | Calls searchWord() function.  nIndex = -1  Prints a note that word was not found and exits the function | Called searchWord() function  nIndex = -1  Printed note that word was not found and exited the function | P |
|  | 4 | User entered a valid string input, and string input exists in the current entry list | strWordToAddTrivia = “Apple Pie”  aEntries[1].answer = “Apple Pie” | Calls searchWord() function.  nIndex = 1  Calls addTrivia() Function | Called searchWord() function  nIndex = 1  Called addTrivia() function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| modifyWord | 1 | User enters an invalid string; empty string | Current entry being modified:  aEntries[2].answer = “Ax” or oneEntry->answer = “Ax”  sModifiedWord = “” | Prints an invalid input note, and asks for another string input | Printed invalid input note and asked for another string input | P |
|  | 2 | User cancels word modification | Current entry being modified:  aEntries[2].answer = “Ax” or oneEntry->answer = “Ax”  Cancel character: “~”  strWordToAddTrivia = “~” | Exits the function | Exited the function | P |
|  | 3 | User entered a valid string input but string input for a new modified word already exists in the current entry list | Current entry being modified:  aEntries[2].answer = “Ax” or oneEntry->answer = “Ax” strWordToAddTrivia = “Ant”  // word exists in index 0  aEntries[0].answer = “Ant” | bUnique = 0  Prints a note that word already exists, and asks user for another string input | bUnique = 0 //index for aEntries  Printed a note that word already exists and asked user for another string input | P |
|  | 4 | User entered a valid string input, and string input for a new modified doesn’t exists yet in the current entry list | Current entry being modified:  aEntries[2].answer = “Ax” or oneEntry->answer = “Ax”  strWordToAddTrivia = “Amazon” | bUnique = -1  Copies the string from strWordToAddTrivia to aEntries[2].answer (oneEntry->answer)  aEntries[2].answer (oneEntry->answer) = “Amazon” | bUnique = -1  aEntries[2].answer (oneEntry->answer) = “Amazon” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| getClueModification | 1 | User enters an invalid string; empty string | sModifiedPhrase = “” | Prints an invalid input note, and asks for another string input | Printed invalid input note, and asked for another input | P |
|  | 2 | User cancels relation or relation value modification for current clue | Cancel character: “~”  sModifiedPhrase = “~” | Exits the function | Exited the function | P |
|  | 3 | User enters a valid string input for relation or relation value | //relation under modification  Current phrase: “Color”  sModifiedPhrase = “Texture” | Copies string from sModifiedPhrase to oneTrivia->relation  oneTrivia->relation = “Texture” | oneTrivia->relation = “Texture” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| modifyClue | 1 | User character input is not within the choices 1, 2 and 0. | cChoice = ‘8’ | Prints an invalid input note, and asks for another input | Printed invalid input note and asked for another input | P |
|  | 2 | User chooses to modify the relation of the current clue | cChoice = ‘1’  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Color”  aEntries[1].clueList[0].relValue = “Golden Brown”  New modified relation member field:  “Texture” | Calls getClueModification() function to get new modified relation member field  aEntries[1].clueList[0].relation = “Texture” | Called getClueModification() function  aEntries[1].clueList[0].relation = “Texture” | P |
|  | 3 | User chooses to modify the relation value of the current clue | cChoice = ‘2’  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Color”  aEntries[1].clueList[0].relValue = “Golden Brown”  New modified relation value member field:  “Crusty” | Calls getClueModification() function to get new modified relation value member field  aEntries[1].clueList[0].relValue = “Crusty” | Called getClueModification() function  aEntries[1].clueList[0].relValue = “Crusty” | P |
|  | 4 | User cancels clue modification of the current clue | cChoice = ‘0’ | Exits the function | Exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| executeModification | 1 | Entered word doesn’t exist in the current entry list | sChosenWord = “qwerty” | Calls searchWord() function.  nIndex = -1  Displays a note that word doesn’t exist.  Exits the function | Called searchWord() function.  nIndex = -1  Displayed a note that word doesn’t exist  Exited the function | P |
|  | 2 | Entered word exists | aEntries[2].answer = “Ax”  sChosenWord = “Ax” | Calls searchWord() function.  nIndex = 1  Asks user which to modify or action: Word, Clue or Cancel modification | Called searchWord() function  nIndex = 1  Asked user which to modify/action: Word, Clue or Cancel modification | P |
|  | 3 | Entered word exists but entered value for cChoice is not in the options | aEntries[2].answer = “Ax”  sChosenWord = “Ax”  cChoice = ‘q’ | Displays an invalid input note, and asks user for another cChoice | Displayed invalid input note and asked user for another cChoice | P |
|  | 4 | Entered word exists and player chose to modify the word of the chosen entry | aEntries[0].answer = “Ant”  sChosenWord = “Ant”  cChoice = ‘W’  Change word to: “Fire Ant” | Calls the modifyWord() function  aEntries[0].answer = “Fire Ant” | Called modifyWord() function  aEntries[0].answer = “Fire Ant” | P |
|  | 5 | Entered word exists and player chose to modify the clues of the chosen entry | aEntries[2].answer = “Ax”  sChosenWord = “Ax”  cChoice = ‘C’  Change relation to: “Used for:” | Calls the modifyEntry() function  aEntries[0].clueList[0].relation = “Used for” | Called modifyEntry() function  aEntries[0].clueList[0].relation = “Used for” | P |
|  | 6 | Entered word exists but chose to exit/cancel modification | aEntries[2].answer = “Ax”  sChosenWord = “Ax”  cChoice = ‘X’ | Exits the function | Exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| clearLastEntry | 1 | Called by the function deleteWord() to delete the last entry | oneEntry->answer = “Kid”  oneEntry->clueList[0].relation = “Species” oneEntry->clueList[0].relValue = “Homo Sapien”  oneEntry->numClues = 1 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | P |
|  | 2 | Entry is already cleared | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | P |
|  | 3 | Entry still contains some garbage values | oneEntry->answer = “”  oneEntry->clueList[0].relation = “Size” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 3 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | oneEntry->answer = “”  oneEntry->clueList[0].relation = “” oneEntry->clueList[0].relValue = “”  oneEntry->numClues = 0 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| executeClueDeletion | 1 | User deletes the last clue of the current chosen entry | nDeleteIndex = 3  \*nNumClues = 4  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder”  aCluesList[2].relation = “Made of”  aCluesList[2].relValue = “Metal”  aCluesList[3].relation = “Powered by”  aCluesList[3].relValue = “Electricity” | \*nNumClues = 3  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder”  aCluesList[2].relation = “Made of”  aCluesList[2].relValue = “Metal” | \*nNumClues = 3  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder”  aCluesList[2].relation = “Made of”  aCluesList[2].relValue = “Metal” | P |
|  | 2 | User deletes the first clue of the current chosen entry | nDeleteIndex = 0  \*nNumClues = 4  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder”  aCluesList[2].relation = “Made of”  aCluesList[2].relValue = “Metal”  aCluesList[3].relation = “Powered by”  aCluesList[3].relValue = “Electricity” | \*nNumClues = 3  aCluesList[0].relation = “Shape”  aCluesList[0].relValue = “Cylinder”  aCluesList[1].relation = “Made of”  aCluesList[1].relValue = “Metal”  aCluesList[2].relation = “Powered by”  aCluesList[2].relValue = “Electricity” | \*nNumClues = 3  aCluesList[0].relation = “Shape”  aCluesList[0].relValue = “Cylinder”  aCluesList[1].relation = “Made of”  aCluesList[1].relValue = “Metal”  aCluesList[2].relation = “Powered by”  aCluesList[2].relValue = “Electricity” | P |
|  | 3 | User deletes a clue in between the first and last clues of the current chosen entry | nDeleteIndex = 2  \*nNumClues = 4  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder”  aCluesList[2].relation = “Made of”  aCluesList[2].relValue = “Metal”  aCluesList[3].relation = “Powered by”  aCluesList[3].relValue = “Electricity” | \*nNumClues = 3  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder  aCluesList[2].relation = “Powered by”  aCluesList[2].relValue = “Electricity” | \*nNumClues = 3  aCluesList[0].relation = “Usage”  aCluesList[0].relValue = “Cleaning”  aCluesList[1].relation = “Shape”  aCluesList[1].relValue = “Cylinder  aCluesList[2].relation = “Powered by”  aCluesList[2].relValue = “Electricity” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| deleteClue | 1 | Entered word doesn’t exist in the current entry list | sWordForClue = “qwerty” | Displays a note that word doesn’t exist in the list and exits the function | Displayed a note that word doesn’t exist and exited the function | P |
|  | 2 | Entered a word exists and number of clues is greater than 1 | sWordForClue = “Ax” or aEntries[2].answer = “Ax” | Asks the user which clue to delete. | Asked the user which clue to delete |  |
|  | 3 | Entered a word exists and number of clues is greater than 1. Then player entered an invalid choice for clue deletion | sWordForClue = “Ax” or aEntries[2].answer = “Ax”  aEntries[2].numClues = 3  nChoice = 5 | Asks user for another clue number choice for clue deletion. | Asked user for another clue number choice for clue deletion | P |
|  | 4 | Entered a word exists and number of clues is greater than 1. Then player entered a valid choice for clue deletion | sWordForClue = “Ax” or aEntries[2].answer = “Ax”  aEntries[2].numClues = 3  nChoice = 3  aEntries[2].clueList[0].relation = “Used for”  aEntries[2].clueList[0].relValue = “Chopping wood”  aEntries[2].clueList[1].relation = “Sharpness”  aEntries[2].clueList[1].relValue = “High”  aEntries[2].clueList[2].relation = “Made of”  aEntries[2].clueList[2].relValue = “Wood and metal” | Calls executeClueDeletion() function.  aEntries[2].numClues = 3  aEntries[2].clueList[0].relation = “Used for”  aEntries[2].clueList[0].relValue = “Chopping wood”  aEntries[2].clueList[1].relation = “Sharpness”  aEntries[2].clueList[1].relValue = “High” | Called executeclueDeletion() function  aEntries[2].numClues = 3  aEntries[2].clueList[0].relation = “Used for”  aEntries[2].clueList[0].relValue = “Chopping wood”  aEntries[2].clueList[1].relation = “Sharpness”  aEntries[2].clueList[1].relValue = “High” | P |
|  | 5 | Entered word exists but it only has 1 clue/trivia left | sWordForClue = “Apple pie” or aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 1  aEntries[2].clueList[0].relation = “Kind of”  aEntries[2].clueList[0].relValue = “Dessert” | Displays a note to the user that there should be at least one trivia per word. | Displayed a note to the user that there should be at least one trivia per word/entry. | P |
|  | 6 | Player cancels clue deletion. | sWordForClue = “~” | Exits the function | Exited the function | P |
|  | 7 | Player entered valid word/entry but decides to cancel clue deletion while in the middle of picking a clue to delete | nChoice = 0 | Exits the function | Exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| viewClues | 1 | Player decides to cancel | strViewClues = “~” | Exits the function | Exited the function | P |
|  | 2 | There are no existing entries yet, and player tried to enter a word | nElem = 0 | Displays a note that there are no entries yet and exits the function | Displayed a note that there are no entries yet and exited the function | P |
|  | 3 | Player entered a word that doesn’t exist yet | strViewClues = “Capybara”  aEntries[0].answer = “Leaves” aEntries[1].answer = “Ant” aEntries[2].answer = “Apple Pie”  aEntries[3].answer = “Car”  aEntries[4].answer = “Zebra”  aEntries[5].answer = “Yoyo”  aEntries[6].answer = “Tea” | Displays a note that word was not found and exits the function | Displayed a note that word was not found and exited the function | P |
|  | 4 | Player entered a word that exists | strViewClues = “Leaves”  aEntries[0].answer = “Leaves” aEntries[1].answer = “Ant” aEntries[2].answer = “Apple Pie”  aEntries[3].answer = “Car”  aEntries[4].answer = “Zebra”  aEntries[5].answer = “Yoyo”  aEntries[6].answer = “Tea” | Calls viewEntry() function and displays all the clues of aEntries[0].answer = “Leaves” | Called viewEntry() function and displayed all the clues of aEntries[0].answer = “Leaves” | P |
|  | 5 | After the clues was displayed, player entered an invalid exit character | cChoice = ‘Q’ | Refreshes the display | Refreshed the display | P |
|  | 6 | After the clues was displayed, player exits the view clues | cChoice = ‘X’ | Exits the function | Exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| swapEntries | 1 | Called by the sortEntries() function | sDestination = &aEntries[1].  sSource = &aEntries[0] | sDestination = &aEntries[0]  sSource = &aEntries[1] | sDestination = &aEntries[0]  sSource = &aEntries[1] | P |
|  | 2 | Current min entry is the minimum (ASCII) for that round | aEntries[0].answer = “Ant”  Min: aEntries[0].answer  sDestination = &aEntries[0]  sSource = &aEntries[0] | aEntries[0].answer = “Ant” | aEntries[0].answer = “Ant” | P |
|  | 3 | Current min entry is not the min ASCII | Min: aEntries[0].answer = “Apple Pie”  aEntries[1].answer = “Ant”  sDestination = &aEntries[0]  sSource = &aEntries[1] | aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie” | aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| import | 1 | File doesn’t exist | fname = “qwerty.txt” | Displays a note that the file doesn’t exist | Displays a note that the file doesn’t exist | P |
|  | 2 | File exists and there is a word in the file that is also in the current entry but user chose to retain the word in the entry. | fname = “SAMPLE-TRIVIA.txt”  cChoice = ‘N’  aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  \*nElem = 3  In File with the same word:  Object: Apple Pie  Made of: Apple and dough  Cooked using: Oven  Other words in file:  Object: Car  Size: Big | Retains the word in the entry and scan the unique entries left in the file.  aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  \*nElem = 4  Added word:  aEntries[1].answer = “Car”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Size”  aEntries[1].clueList[0].relValue = “Big” | Retained the word in the entry and scanned the unique entries left in the file.  aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 1  aEntries[2].clueList[0].relation = “Kind of”  aEntries[2].clueList[0].relValue = “Dessert”  \*nElem = 4  Added word:  aEntries[1].answer = “Car”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Size”  aEntries[1].clueList[0].relValue = “Big” | P |
|  | 3 | File exists and there is a word in the file that is also in the current entry but user chose to overwrite the word in the entry with the word from the file. | fname = “SAMPLE-TRIVIA.txt”  cChoice = ‘Y’  aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  \*nElem = 3  In File with the same word:  Object: Apple Pie  Made of: Apple and dough  Cooked using: Oven  Other words in file:  Object: Car  Size: Big | aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 2  aEntries[1].clueList[0].relation = “Made of”  aEntries[1].clueList[0].relValue = “Apple and dough”  aEntries[1].clueList[1].relation = “Cooked using”  aEntries[1].clueList[1].relValue = “Oven”  \*nElem = 4  Added word:  aEntries[1].answer = “Car”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Size”  aEntries[1].clueList[0].relValue = “Big” | aEntries[1].answer = “Apple pie”  aEntries[1].numClues = 2  aEntries[2].clueList[0].relation = “Made of”  aEntries[2].clueList[0].relValue = “Apple and dough”  aEntries[2].clueList[1].relation = “Cooked using”  aEntries[2].clueList[1].relValue = “Oven”  \*nElem = 4  Added word:  aEntries[1].answer = “Car”  aEntries[1].numClues = 1  aEntries[1].clueList[0].relation = “Size”  aEntries[1].clueList[0].relValue = “Big” | P |
|  | 4 | Either the entries from files are unique for the current entries or there are no entries yet in the program | fname = “SAMPLE-TRIVIA2.txt”  In File:  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Object: Ax  Usage: Wood cutting | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting” | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting” | P |
|  | 5 | Number of entries in the program reached its max capacity | \*pElem = 150  fname = “LAST.txt”  Last entry in program from file:  aEntries[149] = “Zebra”  aEntries[149].numClues = 1  aEntries[149].clueList[0].relation = “Color”  aEntries[149].clueList[0].relValue = “Black and white”  Remaining in file LAST.txt:  Object: zzz  Meaning: Sleeping | aEntries[149] = “Zebra”  aEntries[149].numClues = 1  aEntries[149].clueList[0].relation = “Color”  aEntries[149].clueList[0].relValue = “Black and white” | aEntries[149] = “Zebra”  aEntries[149].numClues = 1  aEntries[149].clueList[0].relation = “Color”  aEntries[149].clueList[0].relValue = “Black and white” | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| initBoard | 1 | The desired number of elements to be generated on the gameboard is greater than the number of elements the current entry has. | nRows = 5  nCols = 6  /\* total number of elements to be used for gameboard is 30 elements but there are only 7 entries\*/  nElem = 7 | 0 | 0 | P |
|  | 2 | The desired number of elements to be generated on the gameboard is less than the number of elements the current entry has. Can be initialized | nRows = 3  nCols = 2  /\* total number of elements to be used for gameboard is 6 elements \*/  nElem = 7  In File: (INIT1.txt)  Object: Leaves  Color: Green  Grows on: Trees  Object: Ant  Kind of: Insect  Object: Apple Pie  Kind of: Dessert  Object: Car  Size: Big  Object: Zebra  Color: Black and white  Object: Yoyo  Used for: Playing  Object: Tea  Helps in: Relaxation | 1  Gameboard contains letters.  Use member field of used entries are updated with the row that they’ve been used | 1  gameboard = {{‘Y’,’T’},{‘C’,’A’},{‘A’,’L’}}  aEntries[0].answer = “Leaves”  aEntries[0].use = 3  aEntries[1].answer = “Ant”  aEntries[1].use = 2  aEntries[2].answer = “Apple Pie”  aEntries[2].use = 3  aEntries[3].answer = “Car”  aEntries[3].use = 2  aEntries[4].answer = “Zebra”  aEntries[4].use = 0  aEntries[5].answer = “Yoyo”  aEntries[5].use = 1  aEntries[6].answer = “Tea”  aEntries[6].use = 1 | P |
|  | 3 | Gameboard can’t be initialize because of the remaining letters are either the same first letters or it cannot provide a unique first letters for all the remaining rows in the gameboard. | nRows = 2  nCols = 3  /\* total number of elements to be used for gameboard is 6 elements \*/  nElem = 7  In File: (INIT2.txt)  Object: Ax  Usage: Wood cutting  Object: Ant  Kind of: Insect  Object: Apple Pie  Kind of: Dessert  Object: Car  Size: Big  Object: Ache  Synonym: Pain  Object: Age  Can be: Counted  Object: Tea  Helps in: Relaxation | 0 | 0 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| play | 1 | Player entered a letter choice that is not in the current row | nCurrentRow = 0  gameboard[0][0] = ‘Q’  gameboard[0][1] = ‘W’  gameboard[0][2] = ‘E’  gameboard[0][3] = ‘R’  cPlyrInpt = ‘T’ | Displays a note that choice doesn’t exist in the current row, and asks user for another letter of choice | Displayed that choice | P |
|  | 2 | Player cancels the game in the middle of entering his letter input | cPlyrInpt = ‘~’ | Exits the function | Exited the function | P |
|  | 3 | Player entered a letter choice that is in the current row | nCurrentRow = 0  gameboard[0][0] = ‘Q’  gameboard[0][1] = ‘W’  gameboard[0][2] = ‘E’  gameboard[0][3] = ‘R’  cPlyrInpt = ‘W’  aEntries[0].answer = “Wheel”  aEntries[0].use = 1  aEntries[0].clueList[0].relation = “Shape”  aEntries[0].clueList[0].relValue = “Circle” | Displays one randomly chosen clue from the chosen letter of entry, and asks the answer of the user. | Displayed: “Shape: Circle”  Asked user for the answer | P |
|  | 4 | Player answered the chosen word correctly | nCurrentRow = 0  gameboard[0][0] = ‘Q’  gameboard[0][1] = ‘W’  gameboard[0][2] = ‘E’  gameboard[0][3] = ‘R’  cPlyrInpt = ‘W’  aEntries[0].answer = “Wheel”  aEntries[0].use = 1  aEntries[0].clueList[0].relation = “Shape”  aEntries[0].clueList[0].relValue = “Circle”  sPlyrAns = “Wheel” | gameboard[0][0] = ‘\*’  nCurrentRow = 1 | gameboard[0][0] = ‘\*’  nCurrentRow = 1 | P |
|  | 5 | Player answered the chosen word wrong | nCurrentRow = 0  gameboard[0][0] = ‘Q’  gameboard[0][1] = ‘W’  gameboard[0][2] = ‘E’  gameboard[0][3] = ‘R’  cPlyrInpt = ‘W’  aEntries[0].answer = “Wheel”  aEntries[0].use = 1  aEntries[0].clueList[0].relation = “Shape”  aEntries[0].clueList[0].relValue = “Circle”  sPlyrAns = “Pizza” | gameboard[0][0] = ‘-’  nPlyrChance = 3  Asks user for another letter choice to answer | gameboard[0][0] = ‘-’  nPlyrChance = 3  Asked user for another letter to answer | P |
|  | 6 | Player answered all the words in the current row of the gameboard wrong | nPlyrChance = 0 | Displays gameover, and exits the function | Displayed gamover and exits | P |
|  | 7 | Player decides to exit the game while in the middle of entering his answer | Cancel Character: “~”  sPlyrAns = “~” | Exits the function | Exited the function | P |
|  | 8 | Player wins | nRows = 2  nCurrentRow = 2 | Displays a congratulations note | Displayed congratulations note | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| maintenance | 1 | Player entered a letter that is not in the options | cChoice = ‘L’ | Displays an invalid input note and asks user for another letter input | Displayed an invalid input note and asked user for another input | P |
|  | 2 | Player decides to exit the maintenance phase | cChoice = ‘X’ | Exits the maintenance() function. | Exited the maintenance() function | P |
|  | 3 | Player chose the Add Word option | cChoice = ‘1’ | Calls the addWord() function | Called the addWord() function | P |
|  | 4 | Player chose the Add Clue option | cChoice = ‘2’ | Calls the addTriviaMenu() function | Called the addTriviaMenu() function | P |
|  | 5 | Player chose the Modify Entry option | cChoice = ‘3’ | Calls the sortEntries() and executeModification() functions | Called the sortEntries() and executeModification() funcitons | P |
|  | 6 | Player chose the Delete Word option | cChoice = ‘4’ | Calls the sortEntries() and deleteWord() functions | Called the sortEntries() and deleteWord() functions | P |
|  | 7 | Player chose the Delete Clue option | cChoice = ‘5’ | Calls the sortEntries() and deleteClue() functions | Called the sortEntries() and deleteClue() functions | P |
|  | 8 | Player chose the View Words option | cChoice = ‘6’ | Calls the sortEntries() and viewWords() functions | Called the sortEntries() and viewWords() funcitons | P |
|  | 9 | Player chose the View Clues option | cChoice = ‘7’ | Calls the sortEntries() and viewClues() functions | Called the sortEntries() and viewClues() funcitons | P |
|  | 10 | Player chose the Export option, and enters a file name | cChoice = ‘8’  sFileName = “SAMPLE-TRIVIA.txt” | Calls the sortEntries() and export() functions | Called the sortEntries() and export() funcitons | P |
|  | 11 | Player chose the Export option, but cancels export. | cChoice = ‘8’  sFileName = “~” | Loops back on asking the user which option/action to be executed in the maintenance phase | Looped back on asking the user which option/action to be executed in the maintenance phase | P |
|  | 12 | Player chose the Import option, and enters a filename | cChoice = ‘9’  sFileName = “SAMPLE-TRIVIA.txt” | Calls the sortEntries() and import() functions | Called the sortEntries() and import() functions | P |
|  | 13 | Player chose the Import option, but cancels import. | cChoice = ‘9’  sFileName = “~” | Loops back on asking the user which option/action to be executed. | Looped back on asking the user which option/action to be executed in the maintenance phase | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| searchWord | 1 | There are no words yet in the entries | nElem = 0  key = “Acrylic” | -1 | -1 | P |
|  | 2 | The word doesn’t exist in the current entry list | nElem = 3  aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie”  aEntries[2].answer = “Ax”  key = “Acrylic” | -1 | -1 | P |
|  | 3 | The word exists in the current entry list | nElem = 3  aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie”  aEntries[2].answer = “Ax”  key = “Ax” | 2 | 2 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| addWord | 1 | Player cancels to add word | strNewWord = “~” | Exits the function, and returns to the maintenance phase | Exited the function | P |
|  | 2 | Player entered a word that already exists in the current entry list. | strNewWord = “Ax”  aEntries[2].answer = “Ax” | Displays a note that word already exists, and exits the function | Displayed a note that word already exists and exited the function | P |
|  | 3 | Player entered a unique entry | \*nElem = 3  aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie”  aEntries[2].answer = “Ax”  strNewWord = “Bulldog” | Copies strNewWord to aEntries[3].answer, and calls addTrivia() function  \* nElem = 4  aEntries[3].answer = “Bulldog” | \* nElem = 4  aEntries[3].answer = “Bulldog”  Called the addTrivia() function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| addTrivia | 1 | Player decides to cancel add trivia, but number of trivia is not at least 1 (numClues = 0) | oneEntry->numClues = 0  oneEntry->clueList[0].relation = “~”  bHasRel = -1 | Displays a note that there should be at least one trivia per word, and then asks user for a relation and relation value again |  | P |
|  | 2 | Player cancels to add trivia, and trivia/clue count is greater than 1 | oneEntry->numClues = 2  oneEntry->clueList[0].relation = “~”  bHasRel = -1 | Exits the function | Exited the function | P |
|  | 3 | Player entered an invalid input for relation member field | oneEntry->clueList[0] .relation= “”  bHasRel = 0 | Displays invalid input note, and asks for an input for the relation member field | Displayed an invalid input note and asked for input for relation member field | P |
|  | 4 | Player entered a valid input for the relation but invalid input for relation value | oneEntry->clueList[0] .relation= “Texture”  bHasRel = 1  oneEntry->clueList[0] .relValue= “”  bHasRelVal = 0 | Displays invalid input note, and asks for an input for the relation value member field to pair with the recent entered relation member field | Displayed an invalid input note and asked for input for relation value member field to pair with the recent entered relation member field | P |
|  | 5 | Player entered a valid input for the relation, then cancels input for relation value | oneEntry->clueList[1] .relation= “Texture”  bHasRel = 1  oneEntry->clueList[1] .relValue= “~”  bHasRelVal = -1 | It will clear the content of the recent relation, and then exits the function.  oneEntry->clueList[1] .relation= “” |  | P |
|  | 6 | Player entered a valid input for both relation and relation value | oneEntry->numClues = 0  oneEntry->clueList[0] .relation= “Texture”  bHasRel = 1  oneEntry->clueList[0] .relValue= “Rough”  bHasRelVal = 1 | oneEntry->numClues = 1  oneEntry->clueList[0] .relation= “Texture”  oneEntry->clueList[0] .relValue= “Rough”  Then asks user for another set of clues | oneEntry->numClues = 1  oneEntry->clueList[0] .relation= “Texture”  oneEntry->clueList[0] .relValue= “Rough”  Then asked user for another set of clues | P |
|  | 7 | Player reached the max capacity of trivia/clue count | oneEntry->numClues = 10 | Displays a note that the entry has reached the maximum capacity of clues, and then exits the function | Displayed a note that entry has reached maximum capacity of clues, and then exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| modifyEntry | 1 | Player decides to cancel modification of clues of one entry | nNumChoice = 0 | Exits the function | Exited the function | P |
|  | 2 | Player enters an invalid integer input; input is not in the range of choices | nNumChoice = 5  oneEntry->numClues = 4 | Displays an invalid input note | Displayed an invalid input note | P |
|  | 3 | Player enters a valid integer input | oneEntry->numClues = 4  nNumChoice = 2 | Calls the modifyClues() function | Called modifyClues() funciton | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| deleteWord | 1 | Entered word to delete doesn’t exists in the current entry list | sWordToDelete = “Carrot”  \*pElem = 3  aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie”  aEntries[2].answer = “Ax” | Displays a note that word entered doesn’t exist | Displayed a note that word doesn’t exist | P |
|  | 2 | Entered word to delete exists in the current entry list | sWordToDelete = “Apple Pie”  \*pElem = 3  aEntries[0].answer = “Ant”  aEntries[1].answer = “Apple Pie”  aEntries[2].answer = “Ax” | Deletes the word “Apple Pie” with all its clues (aEntries[1].answer).  \*pElem = 2  aEntries[0].answer = “Ant” aEntries[1].answer = “Ax” | Deleted the word “Apple Pie” with all its clues  (aEntries[1].answer).  \*pElem = 2  aEntries[0].answer = “Ant” aEntries[1].answer = “Ax” | P |
|  | 3 | Player decides to cancel word deletion | sWordToDelete = “~” | Exits the function | Exited the function | P |
|  | 4 | There are no words to delete because the list of entries is empty | \*pElem = 0 | Displays a note that there are no entries/words left to delete, and then exits the function | Displayed a note that there are no entries/words left to delete and exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| viewWords | 1 | Player tries to enter previous character even though it is the first word/entry being displayed | e = 0  cChoice = ‘<’ | Refreshes the current displayed entry/word | Refreshed the current displayed entry/word | P |
|  | 2 | Player tries to enter next character even though it is the last word/entry being displayed | nElem = 3  nLastElem = 2  e = 2  cChoice = ‘>’ | Refreshes the current displayed entry/word | Refreshed the current displayed entry/word | P |
|  | 3 | Player entered an invalid input or character choice | cChoice = ‘+’ | Refreshes the current displayed entry/word | Refreshed the current displayed entry/word | P |
|  | 4 | Player entered a valid input or character choice | nElem = 3  nLastElem = 2  e = 0  cChoice = ‘>’ | e = 1  Displays the entry/word of aEntries[1] | e = 1  Displayed the entry/word of aEntries[1] | P |
|  | 5 | Player decides to exit the viewing of words/entries | cChoice = ‘X’ | Exits the function | Exited the function | P |
|  | 6 | There are no existing entries yet | nElem = 0 | Displays a note that there are no existing entries yet | Displayed a note that there are no existing entries yet | P |
|  |  | There are no existing entries yet and player tries to enter next/previous characters | nElem = 0  cChoice = ‘>’ | Refreshes the current display | Refreshed the current display | P |
|  | 7 | There are no exsiting entries yet, and player exits the view clues | nElem = 0  cChoice = ‘X’ | Exits the function | Exited the function | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| export | 1 | File doesn’t exist yet | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting” | Creates the file: (EXPORT1.txt)  To be printed in file:  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Object: Ax  Usage: Wood cutting | Created the file: (EXPORT1.txt)  Printed in file:  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Object: Ax  Usage: Wood cutting | P |
|  | 2 | File already exists and overwrites a new list of entries in it | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[1].clueList[1].relation = “Made of”  aEntries[1].clueList[1].relValue = “Apple and dough”  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting”  Exisiting File: (EXPORT2.txt)  Object: Balloon  Made of: Plastic  Contains: Air or Gas | Overwriten file: (EXPORT2.txt)  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Made of: Apple and dough  Object: Ax  Usage: Wood cutting | Overwriten file: (EXPORT2.txt)  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Made of: Apple and dough  Object: Ax  Usage: Wood cutting | P |
|  | 3 | File already exists and the exported entries has no changes | aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Taste”  aEntries[1].clueList[0].relValue = “Sweet”  aEntries[1].clueList[1].relation = “Made of”  aEntries[1].clueList[1].relValue = “Apple and dough”  aEntries[2].answer = “Ax”  aEntries[2].clueList[0].relation = “Usage”  aEntries[2].clueList[0].relValue = “Wood cutting”  In File: (EXPORT2.txt)  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Made of: Apple and dough  Object: Ax  Usage: Wood cutting | In File: (EXPORT2.txt)  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Made of: Apple and dough  Object: Ax  Usage: Wood cutting | In File: (EXPORT2.txt)  Object: Ant  Kind of: Insect  Object: Apple Pie  Taste: Sweet  Made of: Apple and dough  Object: Ax  Usage: Wood cutting | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Descriptions** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
| sortEntries | 1 | There are no entries to sort | nElem = 0 | Exits the function | Exited the function | P |
|  | 2 | Entries are already sorted | nElem = 7  aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  aEntries[1]. numClues = 1  aEntries[2].answer = “Car”  aEntries[2].clueList[0].relation = “Size”  aEntries[2].clueList[0].relValue = “Big”  aEntries[2]. numClues = 1  aEntries[3].answer = “Leaves”  aEntries[3].clueList[0].relation = “Color”  aEntries[3].clueList[0].relValue = “Green”  aEntries[3].clueList[1].relation = “Grows on”  aEntries[3].clueList[1].relValue = “Trees”  aEntries[3].numClues = 2  aEntries[4].answer = “Tea”  aEntries[4].clueList[0].relation = “Helps in”  aEntries[4].clueList[0].relValue = “Relaxation”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1 | nElem = 7  aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  aEntries[1]. numClues = 1  aEntries[2].answer = “Car”  aEntries[2].clueList[0].relation = “Size”  aEntries[2].clueList[0].relValue = “Big”  aEntries[2]. numClues = 1  aEntries[3].answer = “Leaves”  aEntries[3].clueList[0].relation = “Color”  aEntries[3].clueList[0].relValue = “Green”  aEntries[3].clueList[1].relation = “Grows on”  aEntries[3].clueList[1].relValue = “Trees”  aEntries[3].numClues = 2  aEntries[4].answer = “Tea”  aEntries[4].clueList[0].relation = “Helps in”  aEntries[4].clueList[0].relValue = “Relaxation”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1 | nElem = 7  aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  aEntries[1]. numClues = 1  aEntries[2].answer = “Car”  aEntries[2].clueList[0].relation = “Size”  aEntries[2].clueList[0].relValue = “Big”  aEntries[2]. numClues = 1  aEntries[3].answer = “Leaves”  aEntries[3].clueList[0].relation = “Color”  aEntries[3].clueList[0].relValue = “Green”  aEntries[3].clueList[1].relation = “Grows on”  aEntries[3].clueList[1].relValue = “Trees”  aEntries[3].numClues = 2  aEntries[4].answer = “Tea”  aEntries[4].clueList[0].relation = “Helps in”  aEntries[4].clueList[0].relValue = “Relaxation”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1 | P |
|  | 3 | Entries are not sorted | nElem = 7  aEntries[0].answer = “Leaves”  aEntries[0].clueList[0].relation = “Color”  aEntries[0].clueList[0].relValue = “Green”  aEntries[0].clueList[1].relation = “Grows on”  aEntries[0].clueList[1].relValue = “Trees”  aEntries[0].numClues = 2  aEntries[1].answer = “Ant”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Insect”  aEntries[1].numClues = 1  aEntries[2].answer = “Apple Pie”  aEntries[2].clueList[0].relation = “Kind of”  aEntries[2].clueList[0].relValue = “Dessert”  aEntries[2]. numClues = 1  aEntries[3].answer = “Car”  aEntries[3].clueList[0].relation = “Size”  aEntries[3].clueList[0].relValue = “Big”  aEntries[3]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[6].answer = “Tea”  aEntries[6].clueList[0].relation = “Helps in”  aEntries[6].clueList[0].relValue = “Relaxation”  aEntries[6]. numClues = 1 | nElem = 7  aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  aEntries[1]. numClues = 1  aEntries[2].answer = “Car”  aEntries[2].clueList[0].relation = “Size”  aEntries[2].clueList[0].relValue = “Big”  aEntries[2]. numClues = 1  aEntries[3].answer = “Leaves”  aEntries[3].clueList[0].relation = “Color”  aEntries[3].clueList[0].relValue = “Green”  aEntries[3].clueList[1].relation = “Grows on”  aEntries[3].clueList[1].relValue = “Trees”  aEntries[3].numClues = 2  aEntries[4].answer = “Tea”  aEntries[4].clueList[0].relation = “Helps in”  aEntries[4].clueList[0].relValue = “Relaxation”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1 | nElem = 7  aEntries[0].answer = “Ant”  aEntries[0].clueList[0].relation = “Kind of”  aEntries[0].clueList[0].relValue = “Insect”  aEntries[0].numClues = 1  aEntries[1].answer = “Apple Pie”  aEntries[1].clueList[0].relation = “Kind of”  aEntries[1].clueList[0].relValue = “Dessert”  aEntries[1]. numClues = 1  aEntries[2].answer = “Car”  aEntries[2].clueList[0].relation = “Size”  aEntries[2].clueList[0].relValue = “Big”  aEntries[2]. numClues = 1  aEntries[3].answer = “Leaves”  aEntries[3].clueList[0].relation = “Color”  aEntries[3].clueList[0].relValue = “Green”  aEntries[3].clueList[1].relation = “Grows on”  aEntries[3].clueList[1].relValue = “Trees”  aEntries[3].numClues = 2  aEntries[4].answer = “Tea”  aEntries[4].clueList[0].relation = “Helps in”  aEntries[4].clueList[0].relValue = “Relaxation”  aEntries[4]. numClues = 1  aEntries[5].answer = “Yoyo”  aEntries[5].clueList[0].relation = “Used for”  aEntries[5].clueList[0].relValue = “Playing”  aEntries[5]. numClues = 1  aEntries[4].answer = “Zebra”  aEntries[4].clueList[0].relation = “Color”  aEntries[4].clueList[0].relValue = “Black and white”  aEntries[4]. numClues = 1 | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **#** | **Description** | **Sample Input Data** | **Expected Output** | **Actual Output** | **P / F** |
|  | 1 |  |  |  |  |  |
|  | 2 |  |  |  |  |  |
|  | 3 |  |  |  |  |  |