**Test Script for Machine Project in CCPROG2**

**(Shopping App)**

1. **loadData**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| loadData | 1 | Users[] and Items[] only initialized and doesn’t contain a value while NumUsers and NumItems are initialized to 0 and the both txt files for items and users have no data. | Users[]: empty  Items[]: empty  \*NumUsers: 0  \*NumItems: 0 | Users[]: empty  Items[]: empty  \*NumUsers: 0  \*NumItems: 0 | Users[]: empty  Items[]: empty  \*NumUsers: 0  \*NumItems: 00 | P |
|  | 2 | Users[] and Items[] only initialized and doesn’t contain a value while NumUsers and NumItems are initialized to 0 and the both txt files for items and users will load a n number of data. | Users[]: empty  Items[]: empty  \*NumUsers: 0  \*NumItems: 0 | Users[]: the contents of the Users.txt are properly loaded  Items[]:the contents of the Items.txt  \*NumUsers: n number of users loaded  \*NumItems: n number of items loaded | Users[]: the contents of the Users.txt are properly loaded  Items[]:the contents of the Items.txt  \*NumUsers: n number of users loaded  \*NumItems: n number of items loaded | P |
|  | 3 | Users[] and Items[] are already initialized with data, and NumUsers and NumItems are non-zero | Users[]: already contains data  Items[]: already contains data  \*NumUsers: non-zero  \*NumItems: non-zero | Users[]: initial data + new unique data found in the txt file  Items[]:initial data + new unique data found in the txt file  \*NumUsers: initial integer + new users found  \*NumItems: initial integer + new items found | Users[]: initial data + new unique data found in the txt file  Items[]:initial data + new unique data found in the txt file  \*NumUsers: initial integer + new users found  \*NumItems: initial integer + new items found | P |

1. **countTransactions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| countTransactions | 1 | NumTransacs is initialized to 0 and loads a n number of transactions from Transactions.txt file | \*NumTransacs: 0 | \*NumTransacs: n number of transactions from the file | \*NumTransacs: n number of transactions from the file | P |
|  | 2 | NumTransacs is initialized to 0 and Transactions.txt file doesn’t have data. | \*NumTransacs: 0 | \*NumTransacs: 0 | \*NumTransacs: 0 | P |
|  | 3 | NumTransacs is initialized to n number of transactions and loads a n number of transactions from Transactions.txt file | \*NumTransacs: n number of transacs | \*NumTransacs: n number of transacs + n number of loaded transacs | \*NumTransacs: n number of transacs + n number of loaded transacs | P |

1. **registerUser**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| registerUser | 1 | Users[] is only initialized but doesn’t contain data and numUsers is initialized to 0 before a user registers | Users[]: empty  numUsers: 0 | Users[]: contains the data entered by the new user  numUsers: 1 | Users[]: contains the data entered by the new user  numUsers: 1 | P |
|  | 2 | Users[] contains existing users and numUsers is initialized a number of users in the system. | Users[]: existing data  numUsers: n number of users | Users[]: existing data + the new user’s data  numUsers: n number of users + 1 | Users[]: existing data + the new user’s data  numUsers: n number of users + 1 | P |
|  | 3 | Users[] already contains the maximum allowed number of users | Users[]: contains the maximum allowed number of users  \*NumUsers: maximum allowed number of users | Users[]: remains unchanged  \*NumUsers: remains unchanged | Users[]: remains unchanged  \*NumUsers: remains unchanged |  |

1. **login**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| login | 1 | Users[] is only initialized but doesn’t contain data and numUsers is initialized to 0 before a user logins | Users[]: empty  numUsers: 0 | Users[]: empty  numUsers: 0 | Users[]: empty  numUsers: 0 | P |
|  | 2 | Users[] contains existing users and numUsers is initialized a number of users in the system. | Users[]: existing data  numUsers: n number of users | Users[]: existing data  numUsers: n number of users | Users[]: existing data  numUsers: n number of users | P |
|  | 3 | Users[] contains multiple users with different usernames and passwords | Users[]: contains multiple users with different usernames and passwords  \*NumUsers: number of users in Users[] | Input: valid username and password  Output: Successful login  Input: invalid username and password  Output: Failed login | Input: valid username and password  Output: Successful login  Input: invalid username and password  Output: Failed login |  |

1. **checkUserID**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| checkUserID | 1 | userID is a valid ID, while Users[] and numUsers are properly initialized | UserID: valid ID  Users[]:existing data  numUsers: n number of users | UserID: valid ID  Users[]:existing data  numUsers: n number of users  @returns 1 | UserID: valid ID  Users[]:existing data  numUsers: n number of users  @returns 1 | P |
|  | 2 | userID is an invalid ID, while Users[] and numUsers are properly initialized | UserID: invalid ID  Users[]:existing data  numUsers: n number of users | UserID: invalid ID  Users[]:existing data  numUsers: n number of users  @returns 0 | UserID: invalid ID  Users[]:existing data  numUsers: n number of users  @returns 0 | P |
|  | 3 | userID is an valid ID, while Users[] and numUsers are empty. | UserID: valid ID  Users[]:empty  numUsers: 0 | UserID: valid ID  Users[]:empty  numUsers: 0  @returns 0 | UserID: valid ID  Users[]:empty  numUsers: 0  @returns 0 | P |

1. **adminMenu**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| adminMenu | 1 | Users[] and Items[] only initialized and doesn’t contain a value while NumUsers and NumItems are initialized to 0 and the both txt files for items and users have no data | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | P |
|  | 2 | Users[] and Items[] only contains valid values while NumUsers and NumItems are initialized to n numbers of items and users | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | P |
|  | 3 | Users[] and Items[] are empty, but NumUsers and NumItems are initialized | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 | P |

1. **calculateTotalSales**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| calculateTotalSales | 1 | Start\_date and end\_date contains valid strings of date in MM/DD/YYYY format | Start\_date: valid string of date  end\_date: valid string of date | Start\_date: valid string of date  end\_date: valid string of date  @returns float value of computed total | Start\_date: valid string of date  end\_date: valid string of date  @returns float value of computed total | P |
|  | 2 | Start\_date and end\_date contains invalid strings of date in MM/DD/YYYY format | Start\_date: invalid string of date  end\_date: invalid string of date | Start\_date: invalid string of date  end\_date: invalid string of date  @returns 0.00 as the computed total | Start\_date: invalid string of date  end\_date: invalid string of date  @returns 0.00 as the computed total | P |
|  | 3 | Start\_date and end\_date are valid strings, but no sales occurred in the given period | Start\_date: "01/01/2023"  end\_date: "01/10/2023" | Output: 0.00 (no sales occurred in the given period) | Output: 0.00 (no sales occurred in the given period) |  |

1. **showSellerSales**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| showSellerSales | 1 | Users[] and Items[] only initialized and doesn’t contain a value while NumUsers and NumItems are initialized to 0 and the both txt files for items and users have no data | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | P |
|  | 2 | Users[] and Items[] only contains valid values while NumUsers and NumItems are initialized to n numbers of items and users | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | P |
|  | 3 | Users[] and Items[] are empty, but NumUsers and NumItems are not 0 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10  No Output | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 No Output | P |

1. **showShopaholics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| showShopaholics | 1 | Users[] and Items[] only initialized and doesn’t contain a value while NumUsers and NumItems are initialized to 0 and the both txt files for items and users have no data | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | Users[]: empty  Items[]: empty  NumUsers: 0  NumItems: 0 | P |
|  | 2 | Users[] and Items[] only contains valid values while NumUsers and NumItems are initialized to n numbers of items and users | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | Users[]: existing data  Items[]: existing data  NumUsers: n number of users  NumItems: n number of items | P |
|  | 3 | Users[] and Items[] are empty, but NumUsers and NumItems are not 0 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10  No Output | Users[]: empty  Items[]: empty  NumUsers: 5  NumItems: 10 No Output | P |

1. **printMainMenu**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| printMainMenu | 1 | The function doesn’t contain any parameter and only asks the user what option it wants to select | No parameters | @returns an integer that represents their choice | @returns an integer that represents their choice | P |

1. **selectUserMenu**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| selectUserMenu | 1 | All structs are loaded with proper data, NumItems and NumUsers have valid value, and currentUserID is the valid ID of the user logged in | Users[]:existing data  Items[]: existing data  Transacs[]:existing data  cartItems[]:existing data  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Users[]:existing data  Items[]: existing data  Transacs[]:existing data  cartItems[]:existing data  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Users[]:existing data  Items[]: existing data  Transacs[]:existing data  cartItems[]:existing data  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | P |
|  | 2 | All structs are initialized but doesn’t contain data. NumItems and NumUsers have valid value, and currentUserID is the valid ID of the user logged in | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | P |
|  | 3 | All structs are loaded with proper data, NumItems and NumUsers have valid values, and currentUserID is an invalid ID | Users[]: existing data  Items[]: existing data  Transacs[]: existing data  cartItems[]: existing data  NumUsers: n number of users  NumItems: n number of items  CurrentUserID: invalid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user |  |

1. **loadCartData**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| loadCartData | 1 | cartItems[] is initialized but doesn’t contain any data, NumItemsInCart is initialized to 0, and currentUserID is a valid ID. The function will then load userid.txt file to read the items in cart by the current user | cartItems[]: empty  currentUserID: valid ID of logged in user  \*NumItemsInCart: 0 | cartItems[]: contains the data loaded from the txt file  currentUserID: valid ID of logged in user  \*NumItemsInCart: n number of items in the cart | cartItems[]: contains the data loaded from the txt file  currentUserID: valid ID of logged in user  \*NumItemsInCart: n number of items in the cart | P |
|  | 2 | All structs are initialized but doesn’t contain data. NumItems and NumUsers have valid value, and currentUserID is the valid ID of the user logged in | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | Items[]: empty  Transacs[]: empty  cartItems[]:empty  NumUsers: n number of users  \*NumItems: n number of items  CurrentUserID: valid ID of logged in user | P |
|  | 3 | All structs are loaded with proper data, currentUserID is an invalid ID, and cartItems[] and NumItemsInCart are initialized | cartItems[]: empty  currentUserID: invalid ID of logged in user  NumItemsInCart: initialized to 0 | cartItems[]: empty  currentUserID: invalid ID of logged in user  NumItemsInCart: 0 | cartItems[]: empty  currentUserID: invalid ID of logged in user  NumItemsInCart: 0 |  |

1. **addNewItem**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| addNewItem | 1 | Adding a new item with a unique product ID | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n number of items + new item added  NumItems: n (current number of items) + 1  sellerID: valid seller ID | Items[]: Contains n number of items + new item added  NumItems: n (current number of items) + 1  sellerID: valid seller ID | P |
|  | 2 | Adding a new item with a product ID that already exists | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | P |
|  | 3 | Seller has reached the maximum number of items they can sell | Items[]: Contains MAX\_ITEMS (maximum number of items) for the sellerID  NumItems: MAX\_ITEMS (current number of items)  sellerID: valid seller ID | Items[]: Contains MAX\_ITEMS (maximum number of items) for the sellerID  NumItems: MAX\_ITEMS (current number of items)  sellerID: valid seller ID | Items[]: Contains MAX\_ITEMS (maximum number of items) for the sellerID  NumItems: MAX\_ITEMS (current number of items)  sellerID: valid seller ID | P |

1. **selectEditStock**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| selectEditStock | 1 | The user selected to replenish a product | Items[]: Contains n proper data of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated quantity of the item  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated quantity of the item  NumItems: n (current number of items)  sellerID: valid seller ID | P |
|  | 2 | The user selected to change price | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated price of the item  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated price of the item  NumItems: n (current number of items)  sellerID: valid seller ID | P |
|  | 3 | The user selected to change item name | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated name of the item  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated name of the item  NumItems: n (current number of items)  sellerID: valid seller ID | P |
|  | 4 | The user selected to change category | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated category of the item  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated category of the item  NumItems: n (current number of items)  sellerID: valid seller ID | P |
|  | 5 | The user selected to change description | Items[]: Contains n number of items  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated description of the item  NumItems: n (current number of items)  sellerID: valid seller ID | Items[]: Contains n proper data of items with updated description of the item  NumItems: n (current number of items)  sellerID: valid seller ID | p |

1. **addToCart**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| addAddToCart | 1 | Add an item to the cart with sufficient quantity | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 1  userID: 1001  cartItems[]:initial data of the cart + newly added item | Items[]: existing data  NumItemsInCart: 1  userID: 1001  cartItems[]:initial data of the cart + newly added item | P |
|  | 2 | Add an item to the cart with insufficient quantity | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | P |
|  | 3 | Add an item to the cart with invalid product ID | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 0  userID: 1001  cartItems[]:initial data of the cart | P |

1. **editCart**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| editCart | 1 | Testing the function with an empty cart and selecting option 4 to finish editing the cart. | cartItems[]: empty array  numItemsInCart: 0  items[]: empty array  NumItems: 0 | cartItems[]: empty array  numItemsInCart: 0  items[]: empty array  NumItems: 0 | cartItems[]: empty array  numItemsInCart: 0  items[]: empty array  NumItems: 0 | P |
|  | 2 | Add an item to the cart with insufficient quantity | Items[]: existing data  NumItemsInCart: 2  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 2  userID: 1001  cartItems[]:initial data of the cart | Items[]: existing data  NumItemsInCart: 2  userID: 1001  cartItems[]:initial data of the cart | P |
|  | 3 | Testing the function with a single item in the cart and selecting option 3 to edit the quantity | cartItems[]:  cartItems[0]: {productID: 1, itemName: "Item 1", category: "Category 1", unitPrice: 10.00, quantity: 2}  numItemsInCart: 1  items[]:  items[0]: {productID: 1, itemName: "Item 1", category: "Category 1", unitPrice: 10.00, quantityAvailable: 5, sellerID: 1001}  NumItems: 1 | cartItems[0].quantity should be updated to a new quantity value, numItemsInCart remains 1, no changes to items[], NumItems remains 1 | cartItems[0].quantity should be updated to a new quantity value, numItemsInCart remains 1, no changes to items[], NumItems remains 1 | P |

1. **removeItemsFromSeller**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| removeItemsFromSeller | 1 | Testing removal of items from a seller with items in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 3  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 3  items[]: data exists  NumItems: 10 | P |
|  | 2 | Testing removal of items from a seller with no items in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]:initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |
|  | 3 | Testing removal of items from a seller that doesn’t exist | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |

1. **removeSpecificItem**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| removeSpecificItem | 1 | Test removing an item that exists in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 4  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 4  items[]: data exists  NumItems: 10 | P |
|  | 2 | Test removing an item that does not exist in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]:initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |
|  | 3 | Test removing an item from an empty cart. | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |

1. **editQuantity**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| editQuantity | 1 | Test removing an item that exists in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 4  items[]: data exists  NumItems: 10 | cartItems[]: the items belonged to the entered seller is removed  numItemsInCart: 4  items[]: data exists  NumItems: 10 | P |
|  | 2 | Test removing an item that does not exist in the cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]:initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |
|  | 3 | Test removing an item from an empty cart | cartItems[]: data exists  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | cartItems[]: initial data is the same  numItemsInCart: 5  items[]: data exists  NumItems: 10 | P |

1. **saveCartData**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| saveCartData | 1 | Test saving cart data to a file for a user with existing cart items | currentUserID: 1001  cartItems: existing data  NumItemsInCart: 2 | currentUserID: 1001  cartItems: existing data  NumItemsInCart: 2 | currentUserID: 1001  cartItems: existing data  NumItemsInCart: 2 | P |
|  | 2 | Test saving cart data to a file for a user with an empty cart (no txt file yet) | currentUserID: 1002  cartItems: existing data  NumItemsInCart: 2 | currentUserID: 1002  cartItems: existing data  NumItemsInCart: 2  File named "1002.txt" created with no content | currentUserID: 1002  cartItems: existing data  NumItemsInCart: 2  File named "1002.txt" created with no content | P |

1. **checkOutAll**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| checkOutAll | 1 | Test with one item in the cart | cartItems[]: One item  numItemsInCart: 1  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | P |
|  | 2 | Test with multiple items in the cart | cartItems[]: five items  numItemsInCart: 5  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 5 | currentUserID: 1002  cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 5 | P |
|  | 3 | Test with empty cart | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | P |

1. **checkOutSpecificSeller**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| checkSpecificSeller | 1 | Test with one item the cart | cartItems[]: One item  numItemsInCart: 1  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | P |
|  | 2 | Test with multiple items in the cart but 3 belongs to the same seller | cartItems[]: five items  numItemsInCart: 5  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 2 items  numItemsInCart: 2  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 3 | cartItems[]: 2 items  numItemsInCart: 2  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 3 | P |
|  | 3 | Test with empty cart | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | P |

1. **checkOutSpecificItem**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| checkSpecificItem | 1 | Test with one item the cart | cartItems[]: One item  numItemsInCart: 1  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | P |
|  | 2 | Test with multiple items in the cart | cartItems[]: five items  numItemsInCart: 5  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 4 items  numItemsInCart: 4  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | cartItems[]: 4 items  numItemsInCart: 4  NumItems: 5  Transacs[]:added proper data  Day, month, and year containing valid string date  NumTransacs: 1 | P |
|  | 3 | Test with empty cart | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | cartItems[]: 0 items  numItemsInCart: 0  NumItems: 5  Transacs[]: empty  Day, month, and year containing valid string date  NumTransacs: 0 | P |

1. **getSellerName**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| getSellerName | 1 | Valid seller ID | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system @returns the name of seller with the corresponding ID | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system @returns the name of seller with the corresponding ID | P |
|  | 2 | Invalid seller ID | SellerID: -1  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system @returns “unknown” | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: correct amount of users in the system @returns “unknown” | P |
|  | 3 | Valid seller ID but numUsers is incorrectly initialized | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: incorrect amount of users in the system | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: incorrect amount of users in the system  @returns different name of a seller | SellerID: 1001  Items[]: initial data  numItems: correct amount of items in the system Users[]: initial data  numUsers: incorrect amount of users in the system  @returns different name of a seller |  |

1. **getBuyerName**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FUNCTION** | **#** | **DESCRIPTION** | **SAMPLE INPUT DATA** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **P/F** |
| getBuyerName | 1 | Valid buyer ID | buyerID: 1001 Users[]: initial data  numUsers: correct amount of users in the system | buyerID: 1001 Users[]: initial data  numUsers: correct amount of users in the system @returns the name of buyer with the corresponding ID | buyerID: 1001 Users[]: initial data  numUsers: correct amount of users in the system @returns the name of buyer with the corresponding ID | P |
|  | 2 | Invalid buyer ID | buyerID: -1 Users[]: initial data  numUsers: correct amount of users in the system | buyerID: -1 Users[]: initial data  numUsers: correct amount of users in the system @returns “unknown” | buyerID: -1 Users[]: initial data  numUsers: correct amount of users in the system @returns “unknown” | P |
|  | 3 | Valid buyer ID but numUsers is incorrectly initialized | buyerID: 1001 Users[]: initial data  numUsers: incorrect amount of users in the system | buyerID: 1001 Users[]: initial data  numUsers: incorrect amount of users in the system @returns different name of a buyer | buyerID: 1001 Users[]: initial data  numUsers: incorrect amount of users in the system @returns different name of a buyer | P |