**PSEUDOCODE**

Structure User:

username: string

password: string

securityAnswer1: string

securityAnswer2: string

Structure Product:

id: integer

productName: string

brandName: string

productCategory: string

buyPrice: float

sellPrice: float

stock: integer

Structure BSTNode:

product: Product

left: BSTNode

right: BSTNode

Function displayInOrder(root):

if root is not NULL:

displayInOrder(root.left)

Print product information of root

displayInOrder(root.right)

Function insertProduct(root, product):

if root is NULL:

Create new BSTNode and assign product to its product field

Set left and right fields of the new node to NULL

else:

if product.id is equal to root.product.id:

Return (product with the same ID already exists)

else if product.id is less than root.product.id:

Recursively call insertProduct with root.left and product

else:

Recursively call insertProduct with root.right and product

Function loadProductsFromFile(root):

Open "products.txt" file for reading

If file cannot be opened:

Print "Failed to open the file."

Return

Define maxLines as 100

Define lineCount as 0

Allocate memory for products array of size maxLines

While lineCount is less than maxLines and not reached end of file:

Read a line from the file and store it in product variable using the specified format

If line is read successfully:

Increment lineCount

Else if end of file is reached:

Break from the loop

Else:

Print "Error reading line"

Break from the loop

Close the file

For each product in products array:

Call insertProduct with root and product

Free the memory allocated for products array

Function displayAllProducts(root):

If root is NULL:

Print "Warehouse is empty!"

Else:

Print product table header

Call displayInOrder with root

Function deleteBST(root):

If root is NULL:

Return

Recursively delete left and right subtrees

Free the memory occupied by root

Function isIdExists(root, id):

If root is NULL:

Return 0 (ID not found)

If id is equal to root.product.id:

Return 1 (ID found)

Else if id is less than root.product.id:

Recursively call isIdExists with root.left and id

Else:

Recursively call isIdExists with root.right and id

Function addNewProduct(product):

Print "Add New Product" header

Read product information from user input and store it in the product structure

Set brandName based on user choice

Set productCategory based on user choice

Print "New product added successfully!"

Function isIdExist(id, filename):

Open the file with filename for reading

If file cannot be opened:

Print error message and exit

Read each existingId from the file

If existingId is equal to id:

Close the file

Return true

Close the file

Return false

Function generateProductId(filename):

Generate a random number id between 1000 and 9999

While isIdExist(id, filename) is true:

Generate a new random number id

Return id

Function writeProductDataToFile(product):

Open "products.txt" file for appending

If file cannot be opened:

Print "Failed to open the file."

Return

Write product information to the file using the specified format

Close the file

Function searchProduct(root, id):

If root is NULL:

Print "Product not found."

Else if id is equal to root.product.id:

Print product information of root

Else if id is less than root.product.id:

Recursively call searchProduct with root.left and id

Else:

Recursively call searchProduct with root.right and id

Function writeInOrder(root, file):

If root is not NULL:

Call writeInOrder with root.left and file

Write product information of root to the file using the specified format

Call writeInOrder with root.right and file

Function updateProductStock(root, id, newStock):

If root is NULL:

Print "Product not found."

Return

If id is equal to root.product.id:

Set root.product.stock to newStock

Print "Product stock updated successfully!"

Open "products.txt" file for reading

Open "temp.txt" file for writing

If any of the files cannot be opened:

Print "Failed to open the file."

Return

Read each line from the file

If currentId is equal to id:

Write updated product information to the temp file

Else:

Write the line to the temp file

Close both files

Remove "products.txt" file

Rename "temp.txt" file to "products.txt"

Else if id is less than root.product.id:

Recursively call updateProductStock with root.left, id, and newStock

Else:

Recursively call updateProductStock with root.right, id, and newStock

Function deleteProductConfirmation(rootPtr, id):

If \*rootPtr is NULL:

Print "Product not found."

Else if id is equal to (\*rootPtr).product.id:

Print product information of \*rootPtr

Print confirmation message to delete the product

Read user choice from input

If choice is 'Y' or 'y':

Set current to \*rootPtr

Set parent to NULL

Set successor to current.right

While successor.left is not NULL:

Set parent to successor

Set successor to successor.left

Set (\*rootPtr).product to successor.product

If parent is not NULL:

Set parent.left to successor.right

Else:

Set current.right to successor.right

Free the memory occupied by successor

Print "Product deleted successfully!"

Open "products.txt" file for reading

Open "temp.txt" file for writing

If any of the files cannot be opened:

Print "Failed to open the file."

Return

Read each line from the file

If currentId is not equal to id:

Write the line to the temp file

Close both files

Remove "products.txt" file

Rename "temp.txt" file to "products.txt"

Else:

Print "Deletion canceled."

Else if id is less than (\*rootPtr).product.id:

Recursively call deleteProductConfirmation with (\*rootPtr).left and id

Else:

Recursively call deleteProductConfirmation with (\*rootPtr).right and id

Function resetPassword(user):

Read new password and confirm password from user

If new password is not equal to confirm password:

Print "Passwords do not match. Please try again."

Repeat the process

Set user.password to new password

Print "Password reset successfully!"

Function generateReportByStockDescending(root, file):

If root is not NULL:

Call generateReportByStockDescending with root.right and file

Write product information of root to the file using the specified format

Call generateReportByStockDescending with root.left and file

Function generateReportByStockAscending(root, file):

If root is not NULL:

Call generateReportByStockAscending with root.left and file

Write product information of root to the file using the specified format

Call generateReportByStockAscending with root.right and file

Function generateReportByStock(root, order):

Open the file based on order ("report\_stock\_most\_to\_least.txt" or "report\_stock\_least\_to\_most.txt") for writing

If file cannot be opened:

Print "Failed to open the file."

Return

Write product table header to the file

If order is 1:

Call generateReportByStockDescending with root and file

Else:

Call generateReportByStockAscending with root and file

Close the file

Function displayMainMenu():

Print main menu options

Function main():

Declare user, product, and root variables

Call loadProductsFromFile with root

Open "users.txt" file for reading

If file cannot be opened:

Print "Failed to open the file."

Return

Read user information from the file and assign it to user structure

Close the file

Read username and password from the user

If username and password match the user structure:

Print "Login successful!"

Else:

Print "Invalid username or password."

Read user choice for password reset

If choice is 'Y' or 'y':

Read security answers from the user

If security answers match the user structure:

Call resetPassword with user

Print "Password reset successfully!"

Else:

Print "Incorrect security answer. Password reset failed."

Else:

Exit the program

Repeat the following until user chooses to exit:

Display the main menu

Read user choice from input

If choice is 1:

Repeat the following until user chooses to stop adding products:

Call addNewProduct with product

Generate unique product id using generateProductId

Call writeProductDataToFile with product

Read user choice to add another product

If choice is 'Y' or 'y':

Continue to the next iteration

Else:

Break from the loop

Else if choice is 2:

Repeat the following until user chooses to go back to the menu:

Display all products using displayAllProducts with root

Read user choice for product operations

If choice is 1:

Repeat the following until user chooses to stop searching:

Read product id to search from the user

Call searchProduct with root and id

Read user choice to search another product

If choice is 'Y' or 'y':

Continue to the next iteration

Else:

Reload products from the file

Break from the loop

Else if choice is 2:

Repeat the following until user chooses to stop updating stock:

Read product id and new stock from the user

Call updateProductStock with root, id, and new stock

Read user choice to update another product

If choice is 'Y' or 'y':

Continue to the next iteration

Else:

Reload products from the file

Break from the loop

Else if choice is 3:

Repeat the following until user chooses to stop deleting products:

Read product id to delete from the user

Call deleteProductConfirmation with root, id

Read user choice to delete another product

If choice is 'Y' or 'y':

Continue to the next iteration

Else:

Reload products from the file

Break from the loop

Else if choice is 0:

Break from the loop and go back to the main menu

Else if choice is 3:

Call generateReportByStock with root and user choice for report order

Print "Report generated successfully!"

Else if choice is 0:

Print "Goodbye!"

Call deleteBST with root

Exit the program