Created → May 10, 2025 Last update → May 19, 2025

PROJECT REPORT

Project Report Online Shopping Cart System (Console Based)

Group Members:

Aeliya Haider(Leader)	-F24CSC003
Ahsan Ali	-S25CSC043
Muskan Khan	-S25CSC017

Course Name:
Programming fundamentals

Instructor Name:
Sumra Khan (Theory)
Sir Farzeen Ali (PF Lab)
Date of presentation:
20/5/2025-Tuesday

Index:

Introduction:
Online Shopping Cart System is a console application (c language) that mimics an online shopping store where users can view items, add/remove them from cart, and checkout with the calculated total. I picked this project because it reinforces fundamental programming principles (such as data structures, file operations, and user input validation) yet is expandable for future development (such as a GUI or database). The application will offer a menu of products, handle cart

operations, verify inputs, and prepare an invoice, giving a working but uncomplicated shopping experience.

6. **Objective**:
The purpose of this project is to create a console-based Online Shopping Cart System in (C Programming Language) whereby users can view products, control their cart, and mimic the checkout process. The application is meant to illustrate fundamental programming principles including data structures, file operation, and user input validation, but still deliver a working and intuitive shopping interface.

7. **Tools Used**:
Programming Language: C
IDE: (code) Dev c++
Operating System: Windows

8. System Requirements:
Hardware: Basic pc/laptop with at least 2 GB Ram Software: Any C compiler(dev c++, Vs code)

9.Program Design:
Problem Statement:
This application addresses the issue of manual cart handling by offering a console based application through which users can navigate products, handle their cart,the checkout-while admins get to update stock. It provides data persistence by storing data using files and checks user inputs for reliability.

Algorithm:
1.Initialization Phase:
The program starts and initializes the necessary structures:
Products [ ] array(pre-populated with product data)
Cart [ ] array (initially empty)
Variables for user input and program control

2. Main loop:
The program enters a continuous loop that will run until the user
selects option
6(exit).

3.Menu Display:
[1] View Products
[2] View Cart
[3] Add to Cart
[4] Remove from Cart
[5] Checkout
[6] Exit

4. User Input:
The program waits for and reads the user's numerical input(1-6).
5. Switch Case processing:
Based on the users choice , the program executes the corresponding
case: Case 1: View Products Iterates through the products[ ] array
For each product, displays:
 1. ID
2. Name
3. Price

Returns to main menu
Case 2: View Cart
Checks if cart [ ] is empty
If not empty display each items:
ID
Name
Quantity
Subtotal(price * quantity)
Calculates and displays grand total
Returns to main menu

Case 3: Add to cart
Prompts user for :
Product ID
Quantity
Validates:
Product ID exists in products []
Quantity is >0 and <= available stock
If item already in cart update quantity

Else:
Add new item to the cart []
Returns to the main menu
Case 4:
Remove From Cart
Prompts the user for product ID
Checks if item exists in cart []
If exists :
Removes item entirely or decrements quantity
Else:
Displays “Item not found in cart”
Returns to main menu

Case 5:
Checkout Calculates total cost of all items in cart
Optionally apply discounts(if implemented)
Generates receipt(displays on console and/or saves to file)
Clears the cart [ ] array
Returns to main menu

Case 6: Exit
Breaks out of main loop
Program terminates
Default case :
Handles invalid input(numbers outside 1- 6 or non-numeric)
Displays “Invalid choice.Try again”

Returns to menu display
Loop continuation:
The loop continues until the user selects option 6
After each case complete(except exit), the program:
Redisplays the menu
Waits for new input
Processes the new choices

Termination:
When user selects 6:
Any cleanup operations execute(if needed)
Program exits gracefully

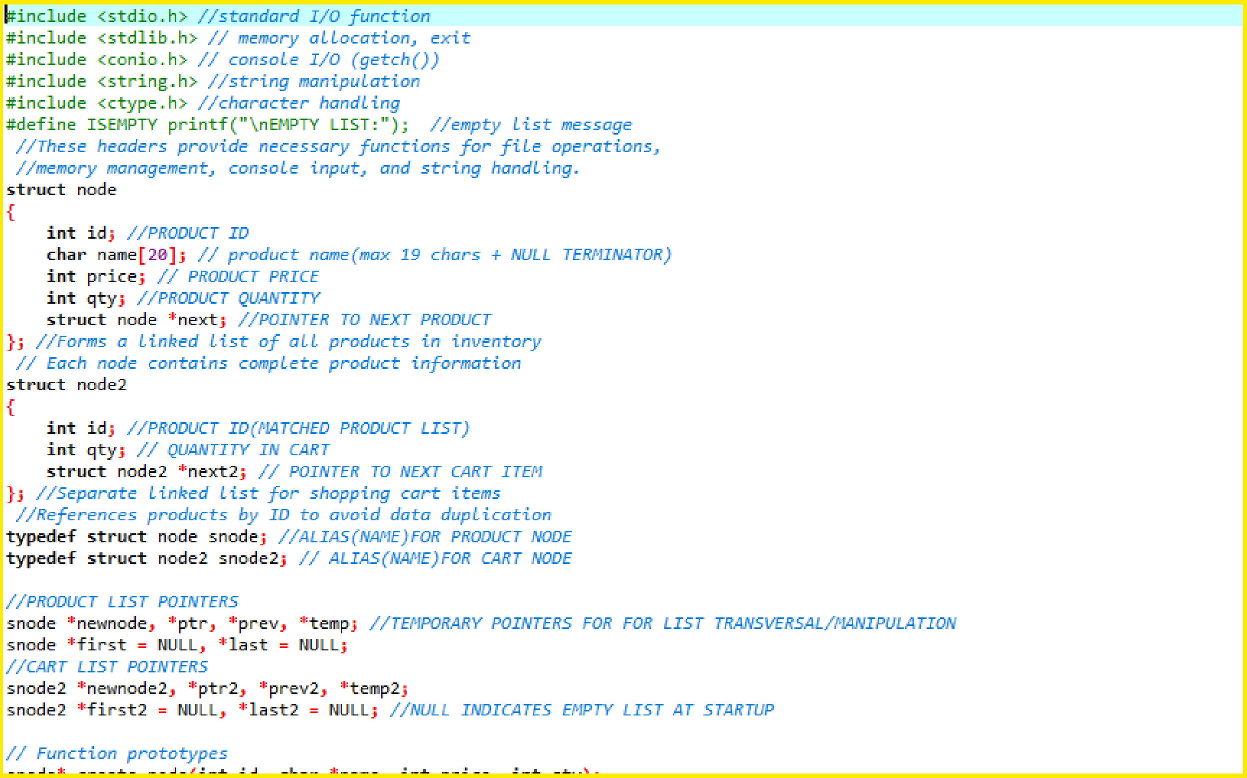
Flowchart:

```
Start
▼
Initialize Product Data (Predefined List)
▼
Display Main Menu:
├── 1. View Products
├── 2. View Cart
├── 3. Add to Cart
├── 4. Remove from Cart
├── 5. Checkout
Start

Online Shopping Cart System in C
▼
Exit
▼
User Input (Choice) ──┐
▼
Switch-Case Structure:
├── 1: Display Product Catalog (Array Traversal) ──┐
├── 2: Display Cart (Array Manipulation) ◄────────┤
├── 3: Add Item to Cart (Quantity Input) ────────┤
├── 4: Remove Item from Cart ────────────────────────┤
├── 5: Checkout (Calculate Total + Receipt) ──────┤
└── 6: Exit Program
▼ │ Repeat Menu Until &quot;Exit&quot; is Selected ◄───────────┘ │ END
```

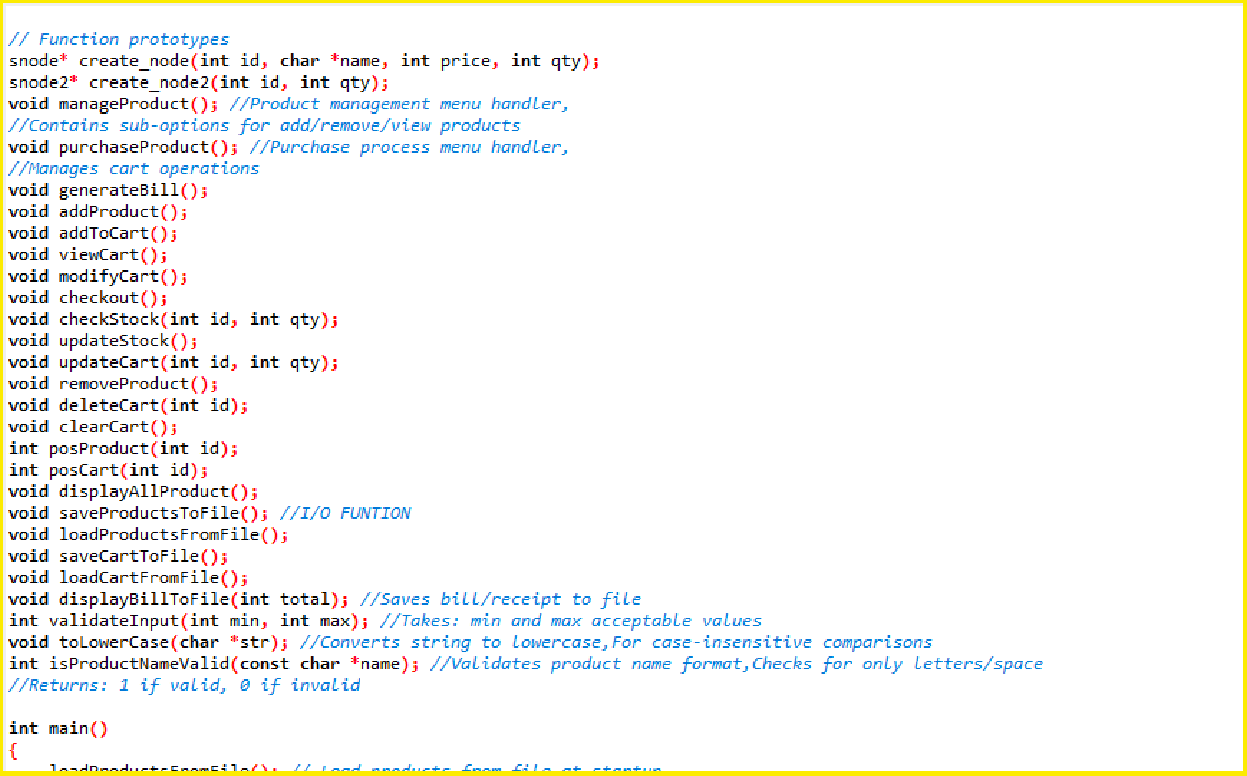
Tools Used

Programming Language: C IDE: (e.g, Embarcadero C++)
Operating System: (e.g., Windows 11)

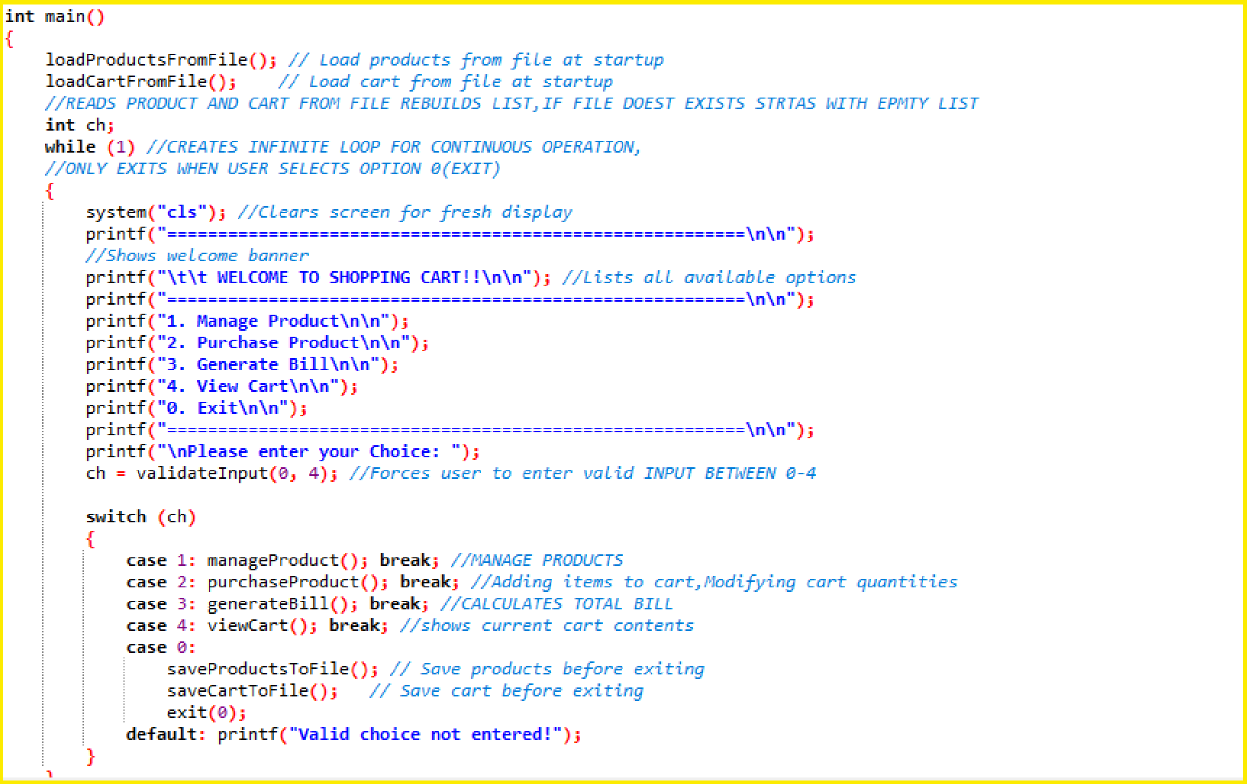
10. Source Code



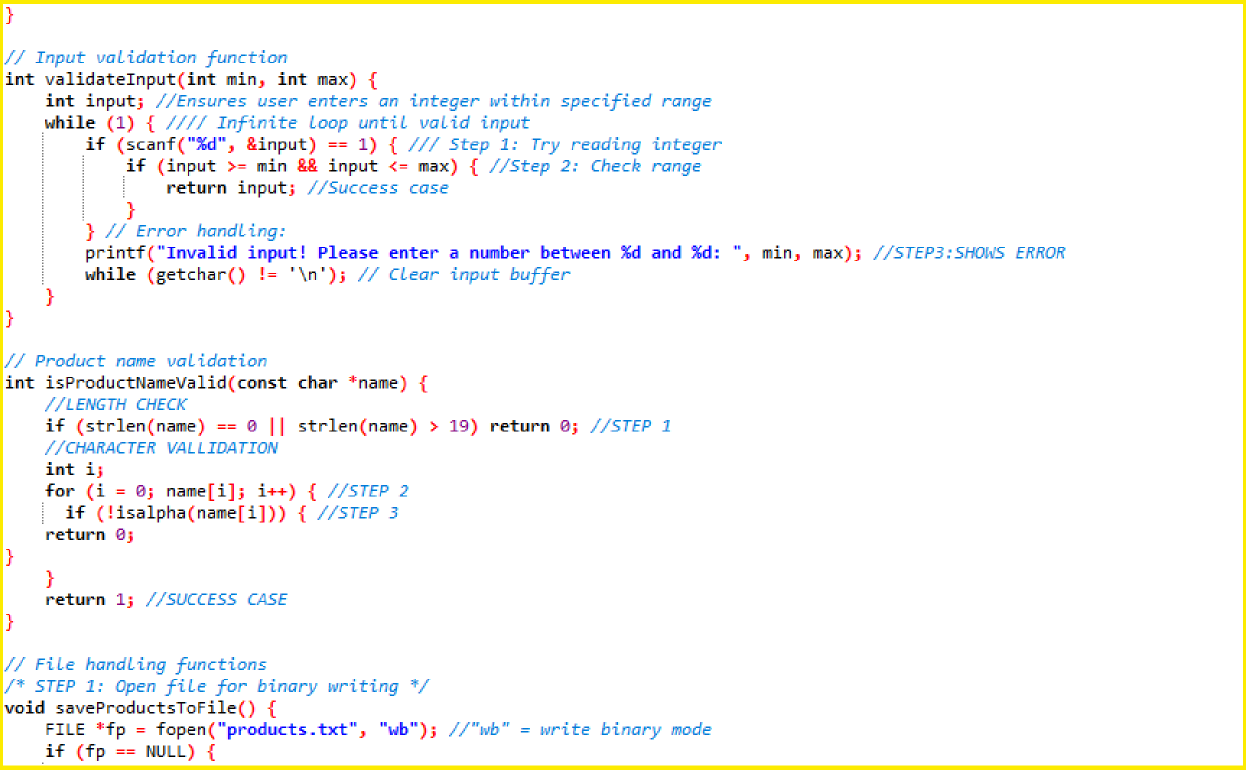
```c
#include <stdio.h> //standard I/O function
#include <stdlib.h> // memory allocation, exit
#include <conio.h> // console I/O (getch())
#include <string.h> //string manipulation
#include <ctype.h> //character handling
#define ISEMPTY printf("\nEMPTY LIST:");  //empty list message
 //These headers provide necessary functions for file operations,
 //memory management, console input, and string handling.
struct node
{
    int id; //PRODUCT ID
    char name[20]; // product name(max 19 chars + NULL TERMINATOR)
    int price; // PRODUCT PRICE
    int qty; //PRODUCT QUANTITY
    struct node *next; //POINTER TO NEXT PRODUCT
}; //Forms a linked list of all products in inventory
 // Each node contains complete product information
struct node2
{
    int id; //PRODUCT ID(MATCHED PRODUCT LIST)
    int qty; // QUANTITY IN CART
    struct node2 *next2; // POINTER TO NEXT CART ITEM
}; //Separate linked list for shopping cart items
 //References products by ID to avoid data duplication
typedef struct node snode; //ALIAS(NAME)FOR PRODUCT NODE
typedef struct node2 snode2; // ALIAS(NAME)FOR CART NODE

//PRODUCT LIST POINTERS
snode *newnode, *ptr, *prev, *temp; //TEMPORARY POINTERS FOR FOR LIST TRANSVERSAL/MANIPULATION
snode *first = NULL, *last = NULL;
//CART LIST POINTERS
snode2 *newnode2, *ptr2, *prev2, *temp2;
snode2 *first2 = NULL, *last2 = NULL; //NULL INDICATES EMPTY LIST AT STARTUP

// Function prototypes
snode* create_node(int id, char *name, int price, int qty);
```



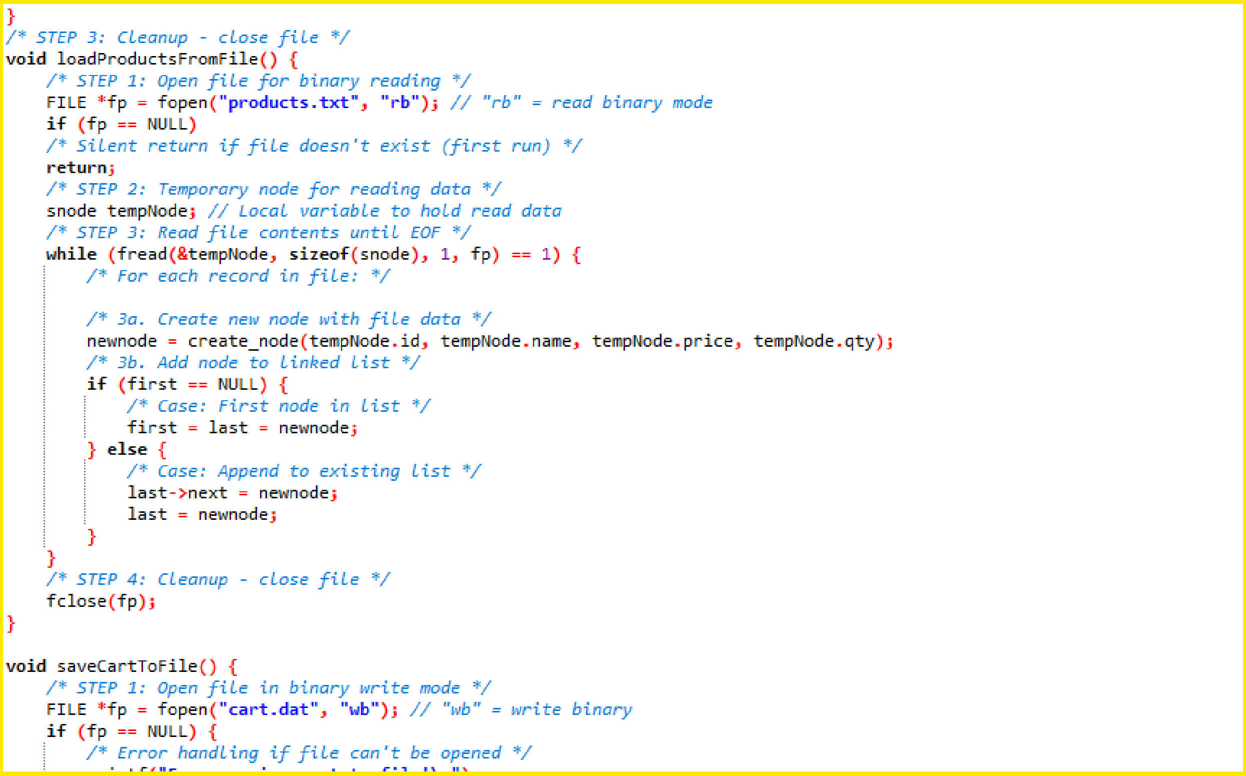
```c
// Function prototypes
snode* create_node(int id, char *name, int price, int qty);
snode2* create_node2(int id, int qty);
void manageProduct(); //Product management menu handler,
//Contains sub-options for add/remove/view products
void purchaseProduct(); //Purchase process menu handler,
//Manages cart operations
void generateBill();
void addProduct();
void addToCart();
void viewCart();
void modifyCart();
void checkout();
void checkStock(int id, int qty);
void updateStock();
void updateCart(int id, int qty);
void removeProduct();
void deleteCart(int id);
void clearCart();
int posProduct(int id);
int posCart(int id);
void displayAllProduct();
void saveProductsToFile(); //I/O FUNTION
void loadProductsFromFile();
void saveCartToFile();
void loadCartFromFile();
void displayBillToFile(int total); //Saves bill/receipt to file
int validateInput(int min, int max); //Takes: min and max acceptable values
void toLowerCase(char *str); //Converts string to lowercase,For case-insensitive comparisons
int isProductNameValid(const char *name); //Validates product name format,Checks for only letters/space
//Returns: 1 if valid, 0 if invalid

int main()
{
```



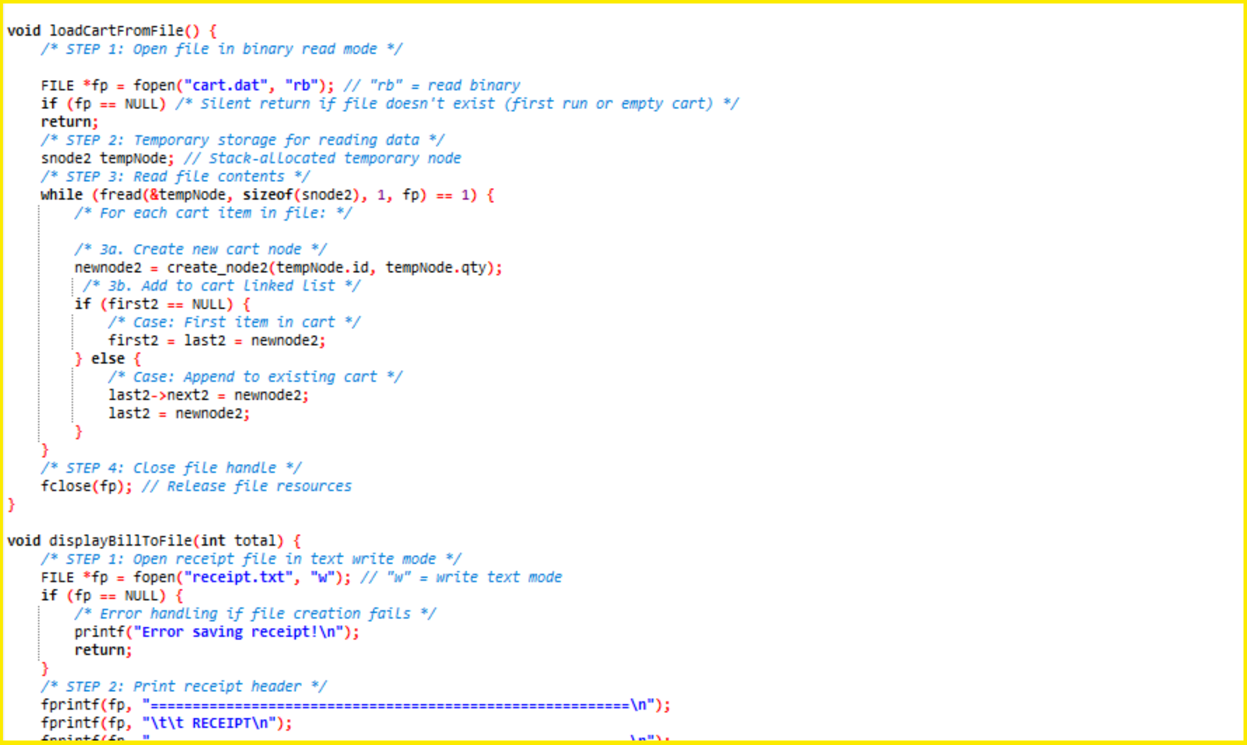
```c
int main()
{
    loadProductsFromFile(); // Load products from file at startup
    loadCartFromFile();    // Load cart from file at startup
    //READS PRODUCT AND CART FROM FILE REBUILDS LIST,IF FILE DOEST EXISTS STRTAS WITH EPMTY LIST
    int ch;
    while (1) //CREATES INFINITE LOOP FOR CONTINUOUS OPERATION,
    //ONLY EXITS WHEN USER SELECTS OPTION 0(EXIT)
    {
        system("cls"); //Clears screen for fresh display
        printf("=========================================================\n\n");
        //Shows welcome banner
        printf("\t\t WELCOME TO SHOPPING CART!!\n\n"); //Lists all available options
        printf("=========================================================\n\n");
        printf("1. Manage Product\n\n");
        printf("2. Purchase Product\n\n");
        printf("3. Generate Bill\n\n");
        printf("4. View Cart\n\n");
        printf("0. Exit\n\n");
        printf("=========================================================\n\n");
        printf("\nPlease enter your Choice: ");
        ch = validateInput(0, 4); //Forces user to enter valid INPUT BETWEEN 0-4

        switch (ch)
        {
            case 1: manageProduct(); break; //MANAGE PRODUCTS
            case 2: purchaseProduct(); break; //Adding items to cart,Modifying cart quantities
            case 3: generateBill(); break; //CALCULATES TOTAL BILL
            case 4: viewCart(); break; //shows current cart contents
            case 0:
                saveProductsToFile(); // Save products before exiting
                saveCartToFile();   // Save cart before exiting
                exit(0);
            default: printf("Valid choice not entered!");
        }
```



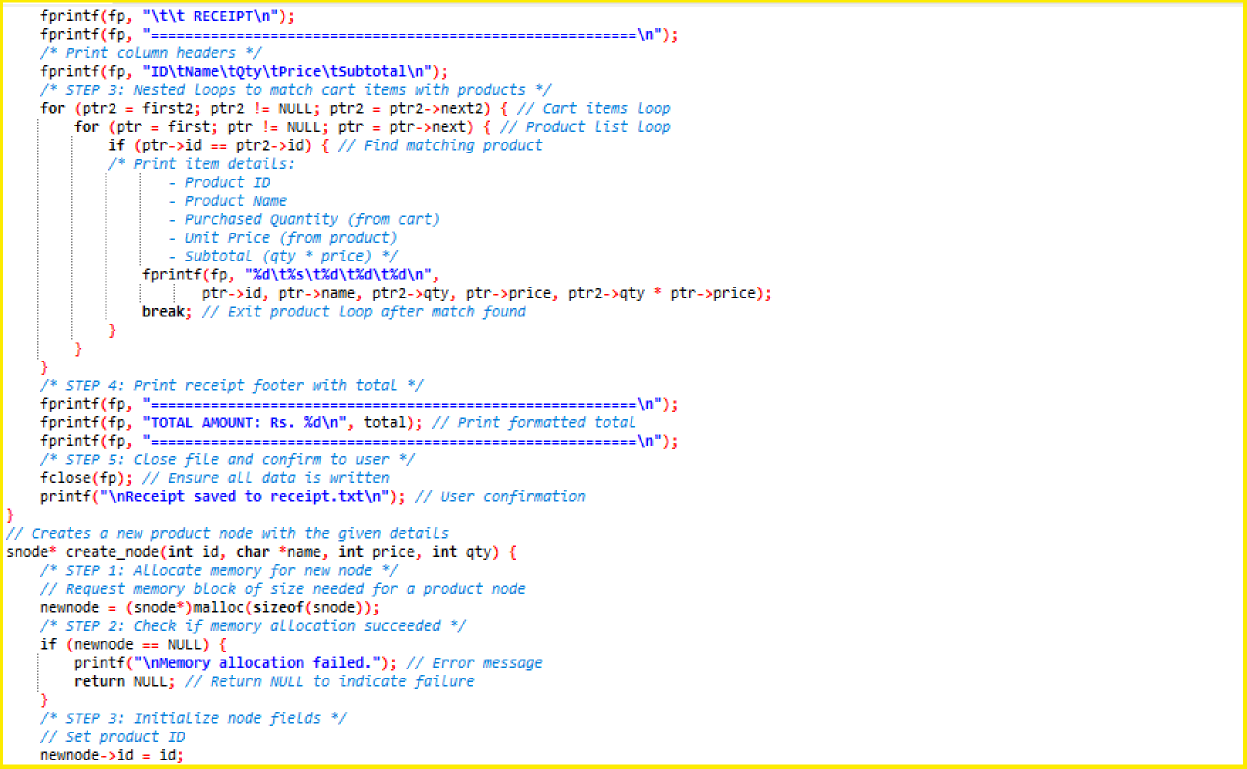
```c
}

// Input validation function
int validateInput(int min, int max) {
    int input; //Ensures user enters an integer within specified range
    while (1) { //// Infinite loop until valid input
        if (scanf("%d", &input) == 1) { /// Step 1: Try reading integer
            if (input >= min && input <= max) { //Step 2: Check range
                return input; //Success case
            }
        } // Error handling:
        printf("Invalid input! Please enter a number between %d and %d: ", min, max); //STEP3:SHOWS ERROR
        while (getchar() != '\n'); // Clear input buffer
    }
}

// Product name validation
int isProductNameValid(const char *name) {
    //LENGTH CHECK
    if (strlen(name) == 0 || strlen(name) > 19) return 0; //STEP 1
    //CHARACTER VALLIDATION
    int i;
    for (i = 0; name[i]; i++) { //STEP 2
      if (!isalpha(name[i])) { //STEP 3
    return 0;
}
    }
    return 1; //SUCCESS CASE
}

// File handling functions
/* STEP 1: Open file for binary writing */
void saveProductsToFile() {
    FILE *fp = fopen("products.txt", "wb"); //"wb" = write binary mode
    if (fp == NULL) {
```



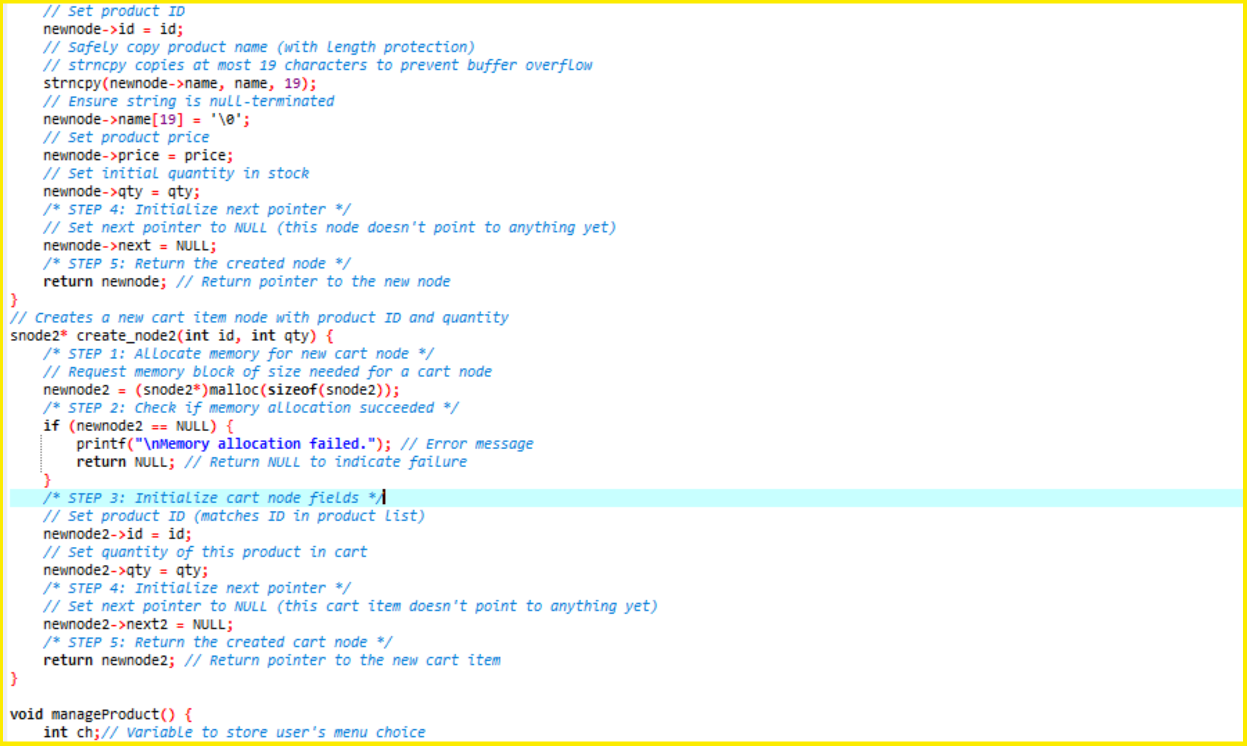
```c
}
/* STEP 3: Cleanup - close file */
void loadProductsFromFile() {
    /* STEP 1: Open file for binary reading */
    FILE *fp = fopen("products.txt", "rb"); // "rb" = read binary mode
    if (fp == NULL)
    /* Silent return if file doesn't exist (first run) */
    return;
    /* STEP 2: Temporary node for reading data */
    snode tempNode; // Local variable to hold read data
    /* STEP 3: Read file contents until EOF */
    while (fread(&tempNode, sizeof(snode), 1, fp) == 1) {
        /* For each record in file: */

        /* 3a. Create new node with file data */
        newnode = create_node(tempNode.id, tempNode.name, tempNode.price, tempNode.qty);
        /* 3b. Add node to linked list */
        if (first == NULL) {
            /* Case: First node in list */
            first = last = newnode;
        } else {
            /* Case: Append to existing list */
            last->next = newnode;
            last = newnode;
        }
    }
    /* STEP 4: Cleanup - close file */
    fclose(fp);
}

void saveCartToFile() {
    /* STEP 1: Open file in binary write mode */
    FILE *fp = fopen("cart.dat", "wb"); // "wb" = write binary
    if (fp == NULL) {
        /* Error handling if file can't be opened */
```



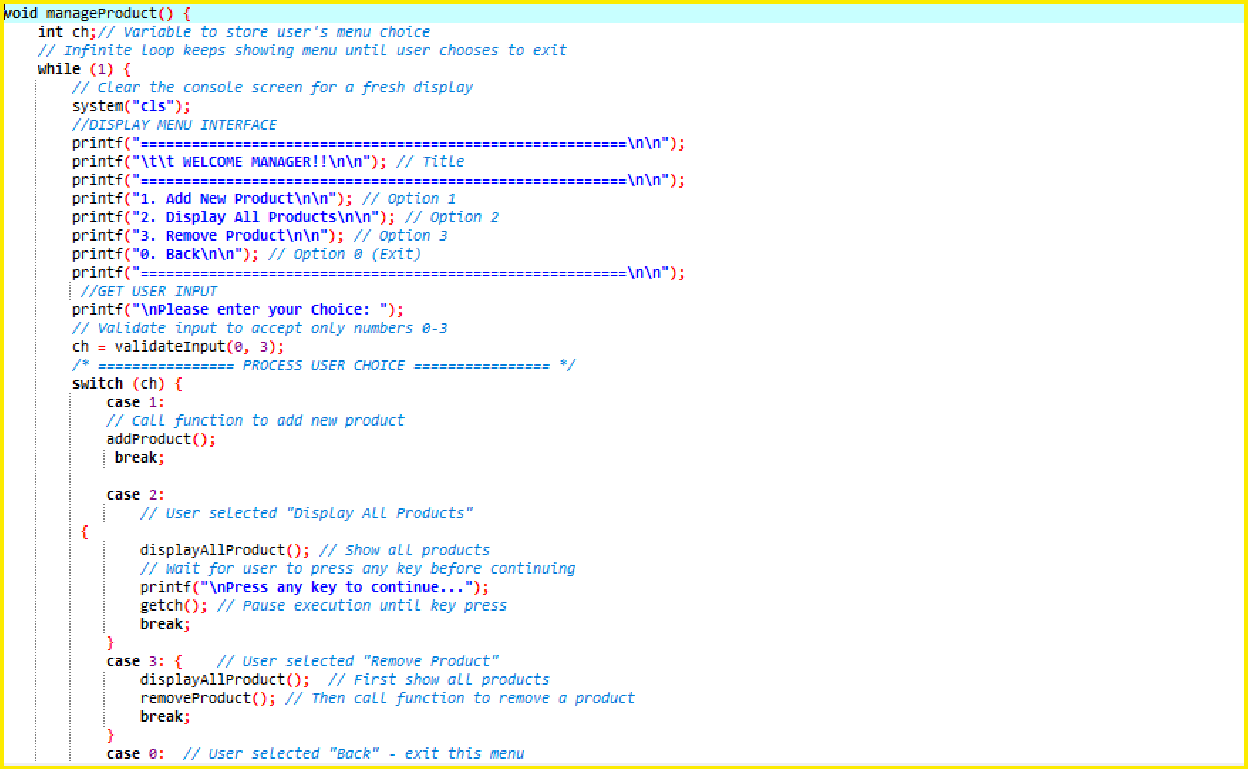
```c
void loadCartFromFile() {
    /* STEP 1: Open file in binary read mode */

    FILE *fp = fopen("cart.dat", "rb"); // "rb" = read binary
    if (fp == NULL) /* Silent return if file doesn't exist (first run or empty cart) */
    return;
    /* STEP 2: Temporary storage for reading data */
    snode2 tempNode; // Stack-allocated temporary node
    /* STEP 3: Read file contents */
    while (fread(&tempNode, sizeof(snode2), 1, fp) == 1) {
        /* For each cart item in file: */

        /* 3a. Create new cart node */
        newnode2 = create_node2(tempNode.id, tempNode.qty);
         /* 3b. Add to cart linked list */
        if (first2 == NULL) {
            /* Case: First item in cart */
            first2 = last2 = newnode2;
        } else {
            /* Case: Append to existing cart */
            last2->next2 = newnode2;
            last2 = newnode2;
        }
    }
    /* STEP 4: Close file handle */
    fclose(fp); // Release file resources
}

void displayBillToFile(int total) {
    /* STEP 1: Open receipt file in text write mode */
    FILE *fp = fopen("receipt.txt", "w"); // "w" = write text mode
    if (fp == NULL) {
        /* Error handling if file creation fails */
        printf("Error saving receipt!\n");
        return;
    }
    /* STEP 2: Print receipt header */
    fprintf(fp, "==========================================================\n");
    fprintf(fp, "\t\t RECEIPT\n");
```



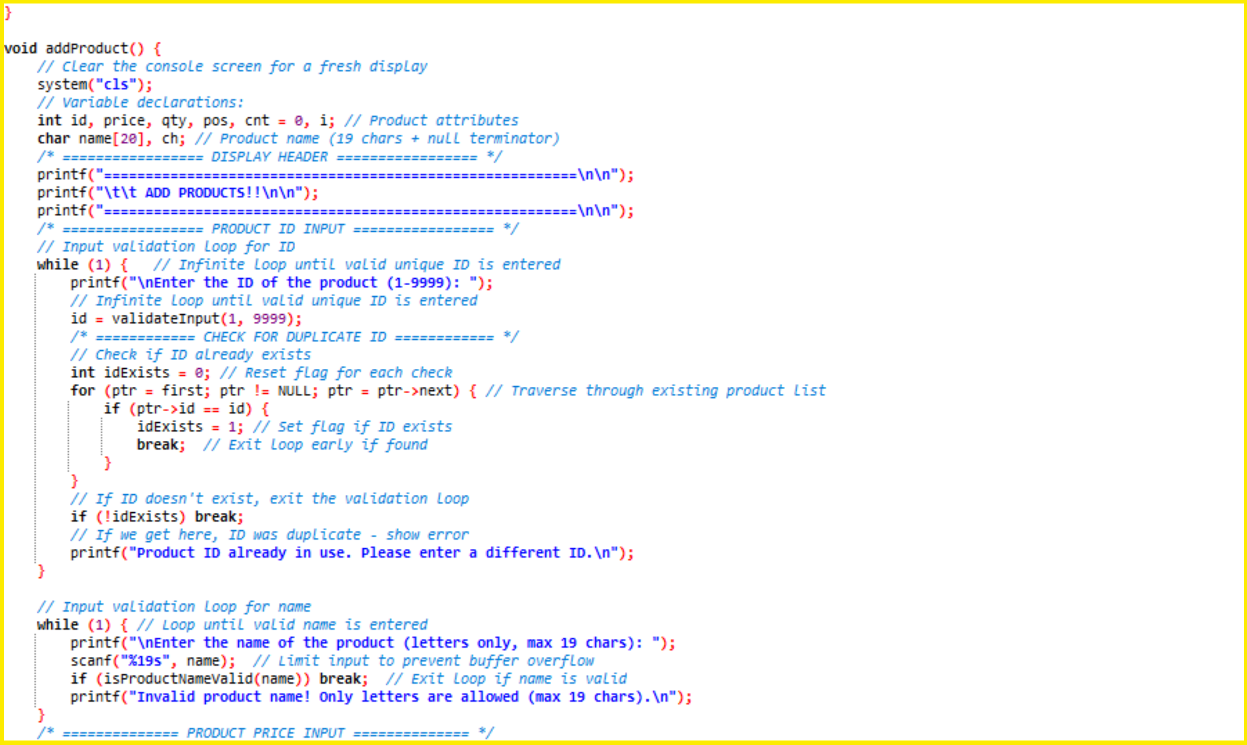
```c
    fprintf(fp, "\t\t RECEIPT\n");
    fprintf(fp, "==========================================================\n");
    /* Print column headers */
    fprintf(fp, "ID\tName\tQty\tPrice\tSubtotal\n");
    /* STEP 3: Nested loops to match cart items with products */
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) { // Cart items loop
        for (ptr = first; ptr != NULL; ptr = ptr->next) { // Product list loop
            if (ptr->id == ptr2->id) { // Find matching product
            /* Print item details:
                   - Product ID
                   - Product Name
                   - Purchased Quantity (from cart)
                   - Unit Price (from product)
                   - Subtotal (qty * price) */
                fprintf(fp, "%d\t%s\t%d\t%d\t%d\n",
                       ptr->id, ptr->name, ptr2->qty, ptr->price, ptr2->qty * ptr->price);
                break; // Exit product loop after match found
            }
        }
    }
    /* STEP 4: Print receipt footer with total */
    fprintf(fp, "==========================================================\n");
    fprintf(fp, "TOTAL AMOUNT: Rs. %d\n", total); // Print formatted total
    fprintf(fp, "==========================================================\n");
    /* STEP 5: Close file and confirm to user */
    fclose(fp); // Ensure all data is written
    printf("\nReceipt saved to receipt.txt\n"); // User confirmation
}
// Creates a new product node with the given details
snode* create_node(int id, char *name, int price, int qty) {
    /* STEP 1: Allocate memory for new node */
    // Request memory block of size needed for a product node
    newnode = (snode*)malloc(sizeof(snode));
    /* STEP 2: Check if memory allocation succeeded */
    if (newnode == NULL) {
        printf("\nMemory allocation failed."); // Error message
        return NULL; // Return NULL to indicate failure
    }
    /* STEP 3: Initialize node fields */
    // Set product ID
    newnode->id = id;
```



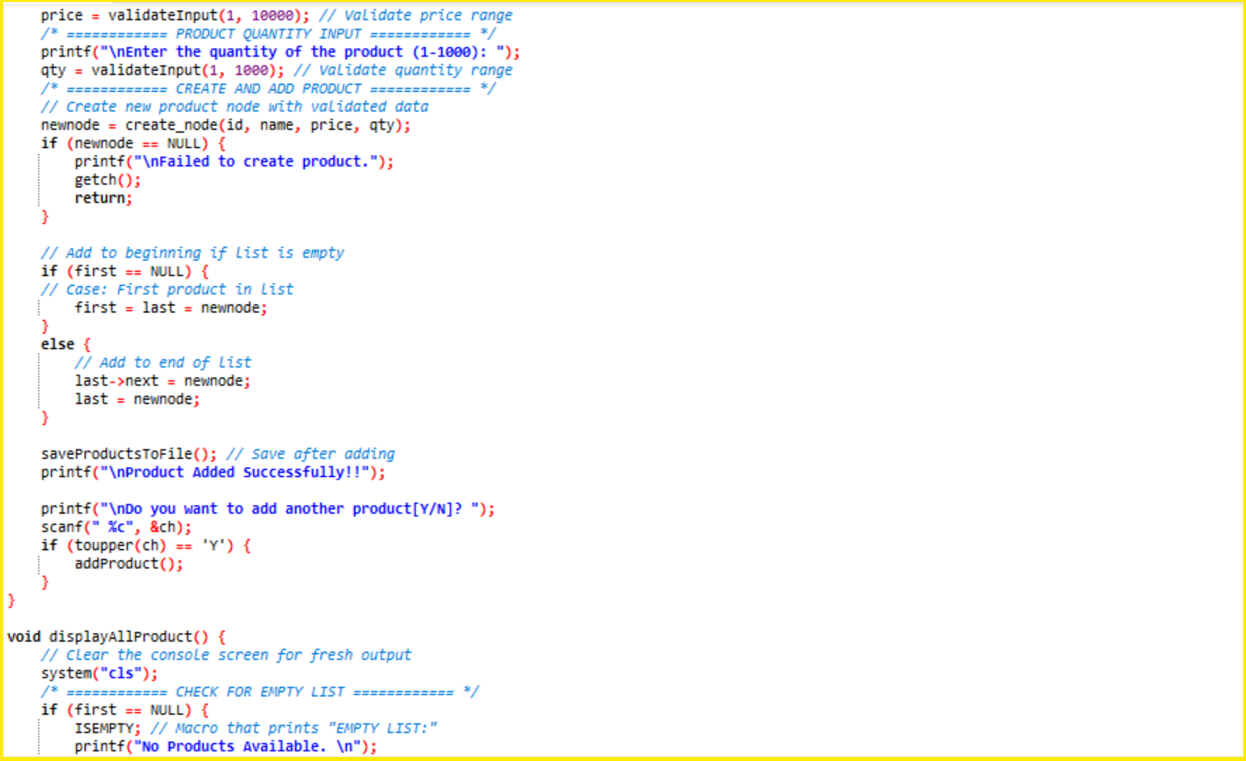
```c
    // Set product ID
    newnode->id = id;
    // Safely copy product name (with length protection)
    // strncpy copies at most 19 characters to prevent buffer overflow
    strncpy(newnode->name, name, 19);
    // Ensure string is null-terminated
    newnode->name[19] = '\0';
    // Set product price
    newnode->price = price;
    // Set initial quantity in stock
    newnode->qty = qty;
    /* STEP 4: Initialize next pointer */
    // Set next pointer to NULL (this node doesn't point to anything yet)
    newnode->next = NULL;
    /* STEP 5: Return the created node */
    return newnode; // Return pointer to the new node
}
// Creates a new cart item node with product ID and quantity
snode2* create_node2(int id, int qty) {
    /* STEP 1: Allocate memory for new cart node */
    // Request memory block of size needed for a cart node
    newnode2 = (snode2*)malloc(sizeof(snode2));
    /* STEP 2: Check if memory allocation succeeded */
    if (newnode2 == NULL) {
        printf("\nMemory allocation failed."); // Error message
        return NULL; // Return NULL to indicate failure
    }
    /* STEP 3: Initialize cart node fields */
    // Set product ID (matches ID in product list)
    newnode2->id = id;
    // Set quantity of this product in cart
    newnode2->qty = qty;
    /* STEP 4: Initialize next pointer */
    // Set next pointer to NULL (this cart item doesn't point to anything yet)
    newnode2->next2 = NULL;
    /* STEP 5: Return the created cart node */
    return newnode2; // Return pointer to the new cart item
}

void manageProduct() {
    int ch;// Variable to store user's menu choice
```



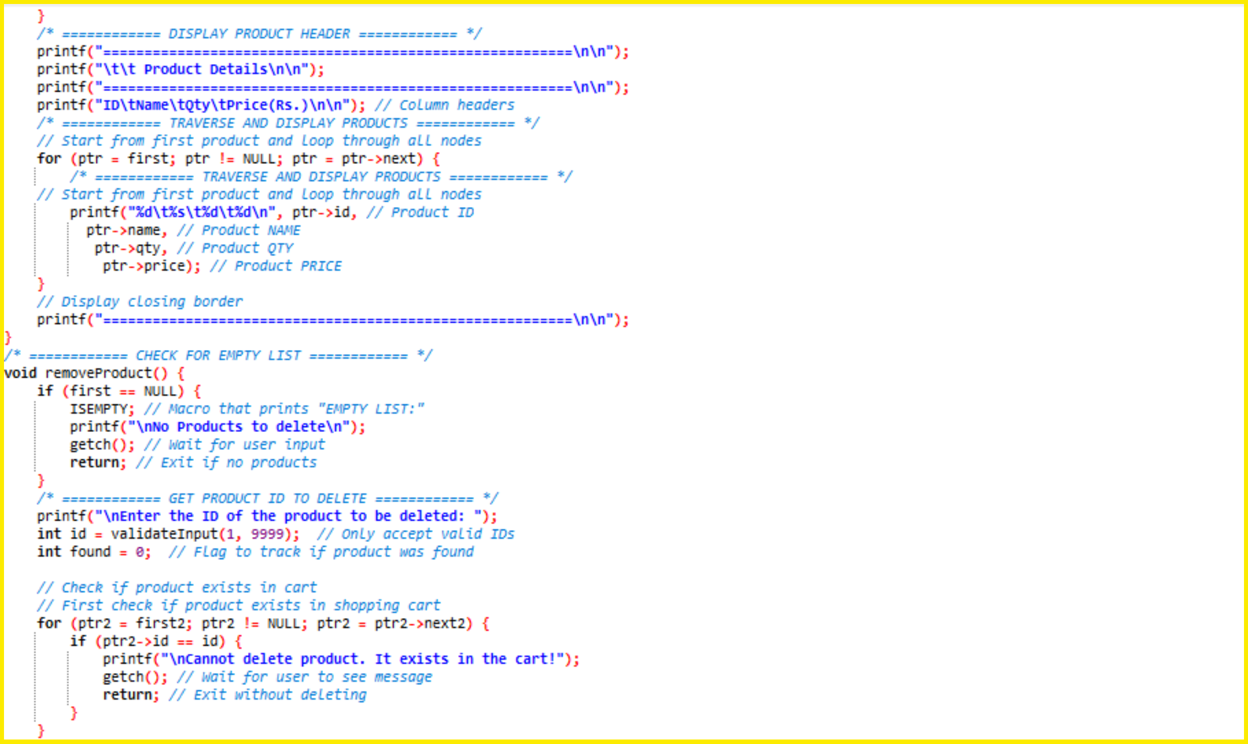
```c
void manageProduct() {
    int ch;// Variable to store user's menu choice
    // Infinite loop keeps showing menu until user chooses to exit
    while (1) {
        // Clear the console screen for a fresh display
        system("cls");
        //DISPLAY MENU INTERFACE
        printf("==========================================================\n\n");
        printf("\t\t WELCOME MANAGER!!\n\n"); // Title
        printf("==========================================================\n\n");
        printf("1. Add New Product\n\n"); // Option 1
        printf("2. Display All Products\n\n"); // Option 2
        printf("3. Remove Product\n\n"); // Option 3
        printf("0. Back\n\n"); // Option 0 (Exit)
        printf("==========================================================\n\n");
         //GET USER INPUT
        printf("\nPlease enter your Choice: ");
        // Validate input to accept only numbers 0-3
        ch = validateInput(0, 3);
        /* =============== PROCESS USER CHOICE =============== */
        switch (ch) {
            case 1:
            // Call function to add new product
            addProduct();
             break;

            case 2:
                // User selected "Display All Products"
         {
                displayAllProduct(); // Show all products
                // Wait for user to press any key before continuing
                printf("\nPress any key to continue...");
                getch(); // Pause execution until key press
                break;
            }
            case 3: {    // User selected "Remove Product"
                displayAllProduct();  // First show all products
                removeProduct(); // Then call function to remove a product
                break;
            }
            case 0:  // User selected "Back" - exit this menu
```



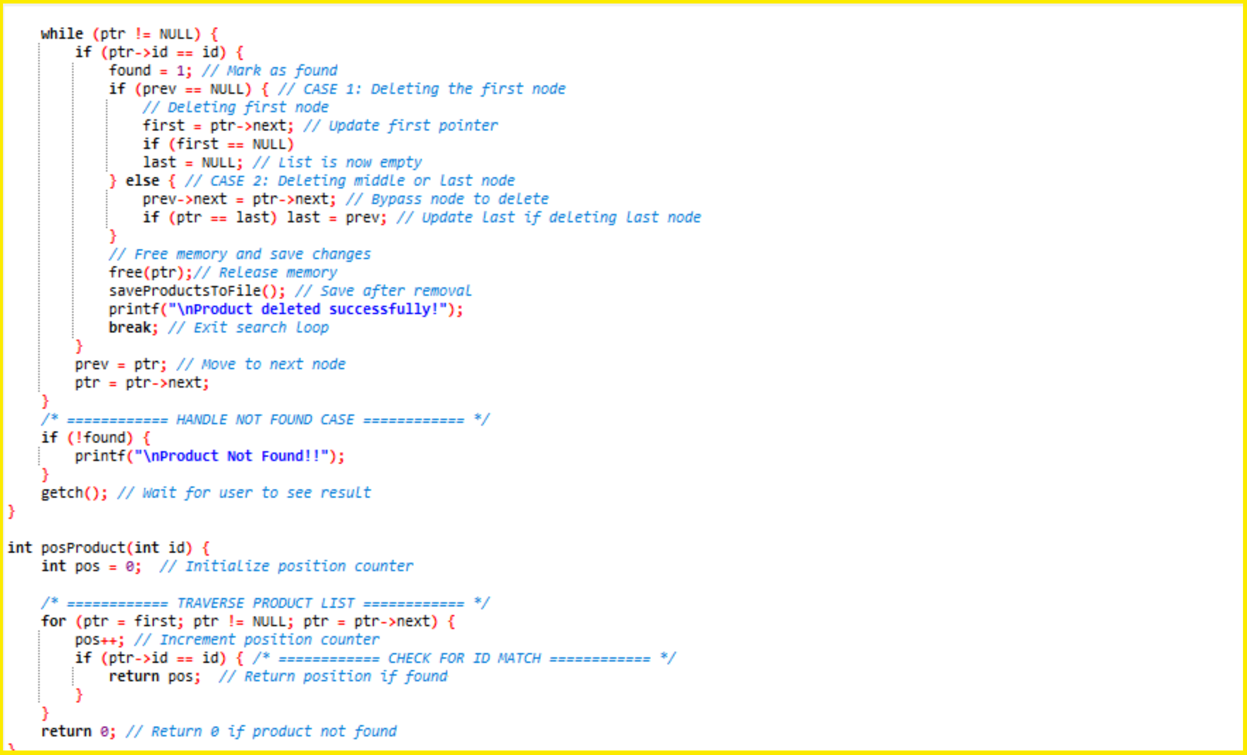
```c
}

void addProduct() {
    // Clear the console screen for a fresh display
    system("cls");
    // Variable declarations:
    int id, price, qty, pos, cnt = 0, i; // Product attributes
    char name[20], ch; // Product name (19 chars + null terminator)
    /* ================ DISPLAY HEADER ================ */
    printf("========================================================\n\n");
    printf("\t\t ADD PRODUCTS!!\n\n");
    printf("========================================================\n\n");
    /* ================ PRODUCT ID INPUT ================ */
    // Input validation loop for ID
    while (1) {   // Infinite loop until valid unique ID is entered
        printf("\nEnter the ID of the product (1-9999): ");
        // Infinite loop until valid unique ID is entered
        id = validateInput(1, 9999);
        /* =========== CHECK FOR DUPLICATE ID =========== */
        // Check if ID already exists
        int idExists = 0; // Reset flag for each check
        for (ptr = first; ptr != NULL; ptr = ptr->next) { // Traverse through existing product list
            if (ptr->id == id) {
                idExists = 1; // Set flag if ID exists
                break;  // Exit loop early if found
            }
        }
        // If ID doesn't exist, exit the validation loop
        if (!idExists) break;
        // If we get here, ID was duplicate - show error
        printf("Product ID already in use. Please enter a different ID.\n");
    }

    // Input validation loop for name
    while (1) { // Loop until valid name is entered
        printf("\nEnter the name of the product (letters only, max 19 chars): ");
        scanf("%19s", name);  // Limit input to prevent buffer overflow
        if (isProductNameValid(name)) break;  // Exit loop if name is valid
        printf("Invalid product name! Only letters are allowed (max 19 chars).\n");
    }
    /* ============= PRODUCT PRICE INPUT ============= */
```



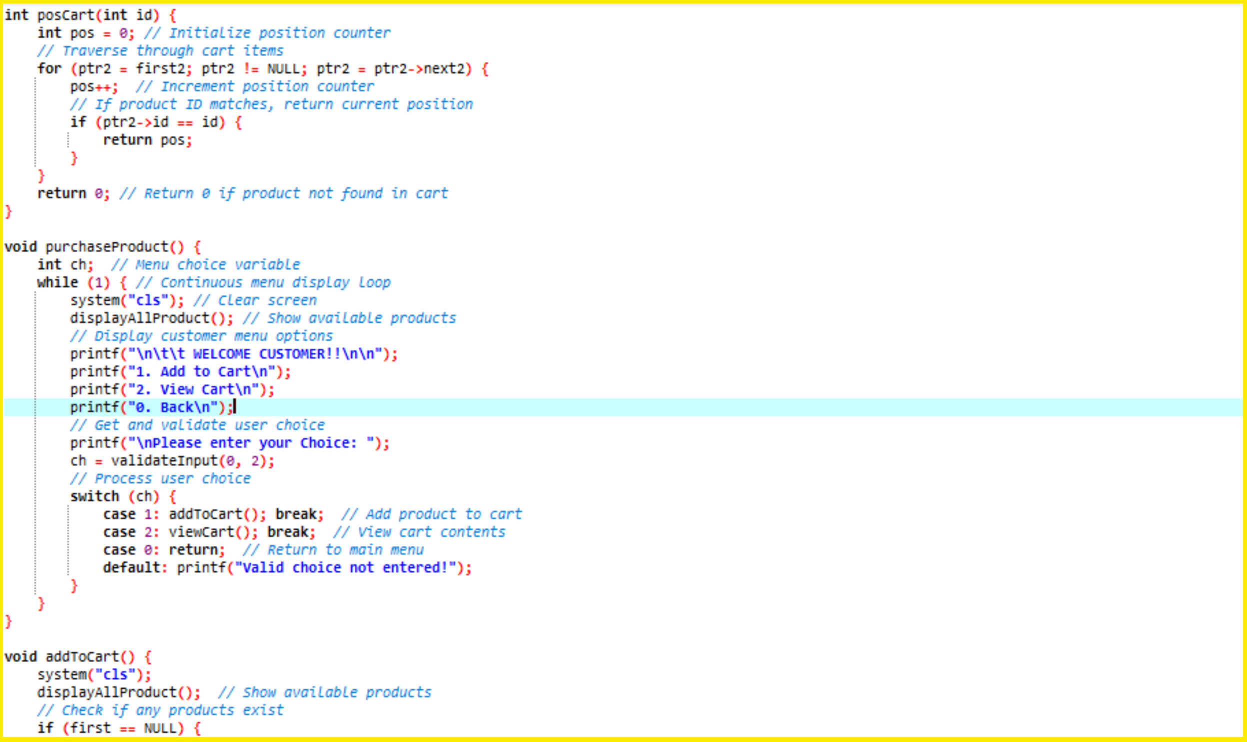
```c
    price = validateInput(1, 10000); // Validate price range
    /* =========== PRODUCT QUANTITY INPUT =========== */
    printf("\nEnter the quantity of the product (1-1000): ");
    qty = validateInput(1, 1000); // Validate quantity range
    /* =========== CREATE AND ADD PRODUCT =========== */
    // Create new product node with validated data
    newnode = create_node(id, name, price, qty);
    if (newnode == NULL) {
        printf("\nFailed to create product.");
        getch();
        return;
    }

    // Add to beginning if list is empty
    if (first == NULL) {
    // Case: First product in list
        first = last = newnode;
    }
    else {
        // Add to end of list
        last->next = newnode;
        last = newnode;
    }

    saveProductsToFile(); // Save after adding
    printf("\nProduct Added Successfully!!");

    printf("\nDo you want to add another product[Y/N]? ");
    scanf(" %c", &ch);
    if (toupper(ch) == 'Y') {
        addProduct();
    }
}

void displayAllProduct() {
    // Clear the console screen for fresh output
    system("cls");
    /* =========== CHECK FOR EMPTY LIST =========== */
    if (first == NULL) {
        ISEMPTY; // Macro that prints "EMPTY LIST:"
        printf("No Products Available. \n");
```



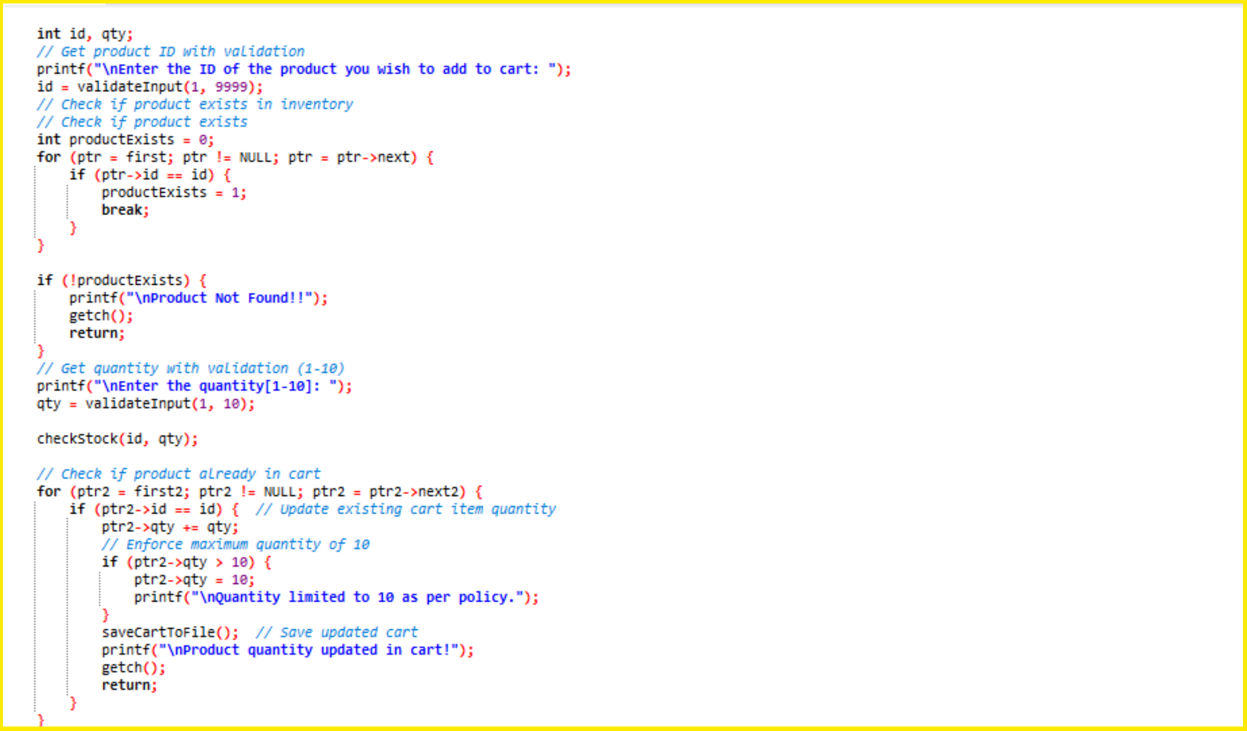
```c
    }
    /* ============ DISPLAY PRODUCT HEADER ============ */
    printf("========================================================\n\n");
    printf("\t\t Product Details\n\n");
    printf("========================================================\n\n");
    printf("ID\tName\tQty\tPrice(Rs.)\n\n"); // Column headers
    /* ============ TRAVERSE AND DISPLAY PRODUCTS ============ */
    // Start from first product and loop through all nodes
    for (ptr = first; ptr != NULL; ptr = ptr->next) {
        /* ============ TRAVERSE AND DISPLAY PRODUCTS ============ */
    // Start from first product and loop through all nodes
        printf("%d\t%s\t%d\t%d\n", ptr->id, // Product ID
          ptr->name, // Product NAME
           ptr->qty, // Product QTY
            ptr->price); // Product PRICE
    }
    // Display closing border
    printf("========================================================\n\n");
}
/* ============ CHECK FOR EMPTY LIST ============ */
void removeProduct() {
    if (first == NULL) {
        ISEMPTY; // Macro that prints "EMPTY LIST:"
        printf("\nNo Products to delete\n");
        getch(); // Wait for user input
        return; // Exit if no products
    }
    /* ============ GET PRODUCT ID TO DELETE ============ */
    printf("\nEnter the ID of the product to be deleted: ");
    int id = validateInput(1, 9999);  // Only accept valid IDs
    int found = 0;  // Flag to track if product was found

    // Check if product exists in cart
    // First check if product exists in shopping cart
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        if (ptr2->id == id) {
            printf("\nCannot delete product. It exists in the cart!");
            getch(); // Wait for user to see message
            return; // Exit without deleting
        }
    }
```



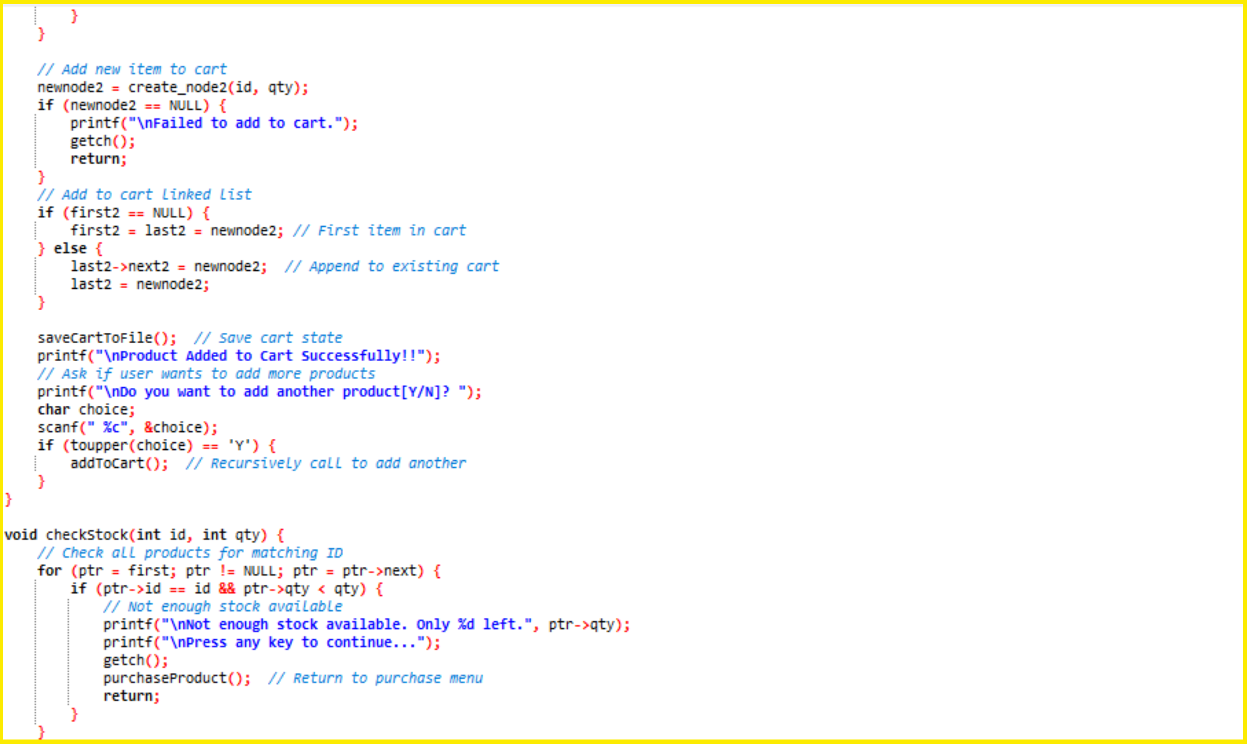
```c
    while (ptr != NULL) {
        if (ptr->id == id) {
            found = 1; // Mark as found
            if (prev == NULL) { // CASE 1: Deleting the first node
                // Deleting first node
                first = ptr->next; // Update first pointer
                if (first == NULL)
                last = NULL; // List is now empty
            } else { // CASE 2: Deleting middle or last node
                prev->next = ptr->next; // Bypass node to delete
                if (ptr == last) last = prev; // Update last if deleting last node
            }
            // Free memory and save changes
            free(ptr);// Release memory
            saveProductsToFile(); // Save after removal
            printf("\nProduct deleted successfully!");
            break; // Exit search loop
        }
        prev = ptr; // Move to next node
        ptr = ptr->next;
    }
    /* ============ HANDLE NOT FOUND CASE ============ */
    if (!found) {
        printf("\nProduct Not Found!!");
    }
    getch(); // Wait for user to see result
}

int posProduct(int id) {
    int pos = 0;  // Initialize position counter

    /* ============ TRAVERSE PRODUCT LIST ============ */
    for (ptr = first; ptr != NULL; ptr = ptr->next) {
        pos++; // Increment position counter
        if (ptr->id == id) { /* ============ CHECK FOR ID MATCH ============ */
            return pos;  // Return position if found
        }
    }
    return 0; // Return 0 if product not found
```



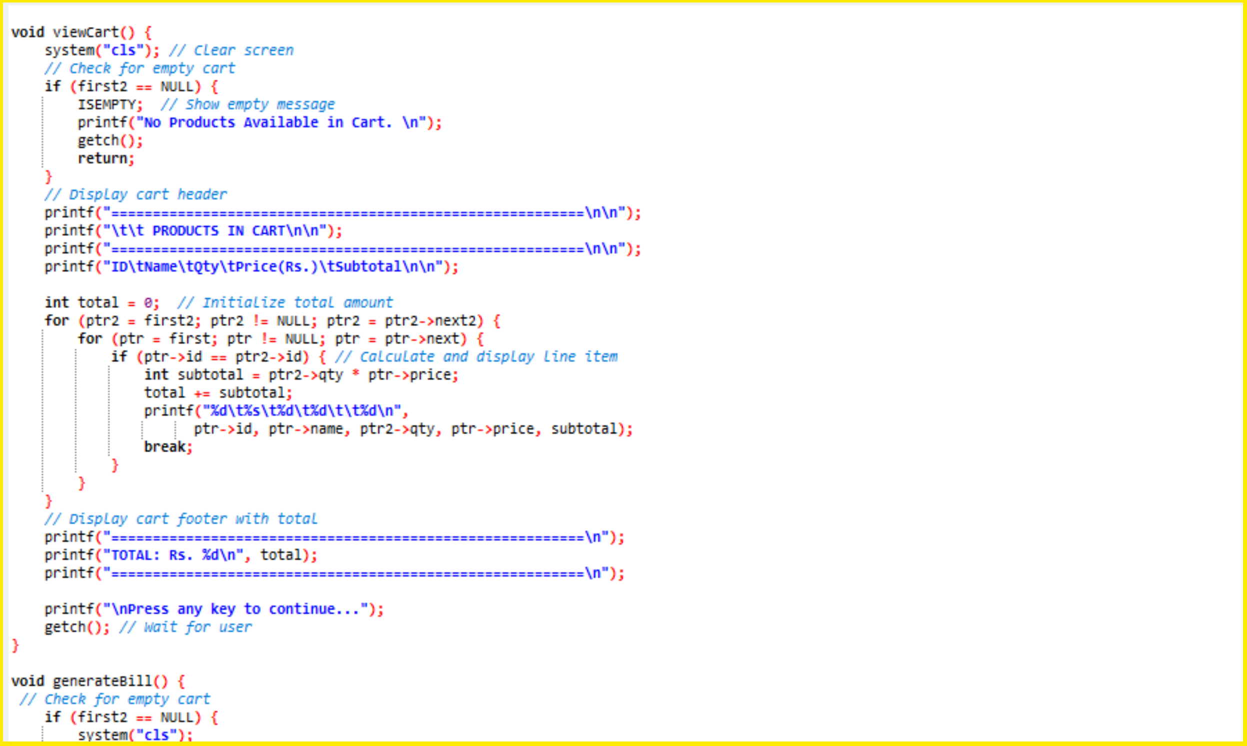
```c
int posCart(int id) {
    int pos = 0; // Initialize position counter
    // Traverse through cart items
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        pos++;  // Increment position counter
        // If product ID matches, return current position
        if (ptr2->id == id) {
            return pos;
        }
    }
    return 0; // Return 0 if product not found in cart
}

void purchaseProduct() {
    int ch;  // Menu choice variable
    while (1) { // Continuous menu display loop
        system("cls"); // Clear screen
        displayAllProduct(); // Show available products
        // Display customer menu options
        printf("\n\t\t WELCOME CUSTOMER!!\n\n");
        printf("1. Add to Cart\n");
        printf("2. View Cart\n");
        printf("0. Back\n");
        // Get and validate user choice
        printf("\nPlease enter your Choice: ");
        ch = validateInput(0, 2);
        // Process user choice
        switch (ch) {
            case 1: addToCart(); break;  // Add product to cart
            case 2: viewCart(); break;  // View cart contents
            case 0: return;  // Return to main menu
            default: printf("Valid choice not entered!");
        }
    }
}

void addToCart() {
    system("cls");
    displayAllProduct();  // Show available products
    // Check if any products exist
    if (first == NULL) {
```



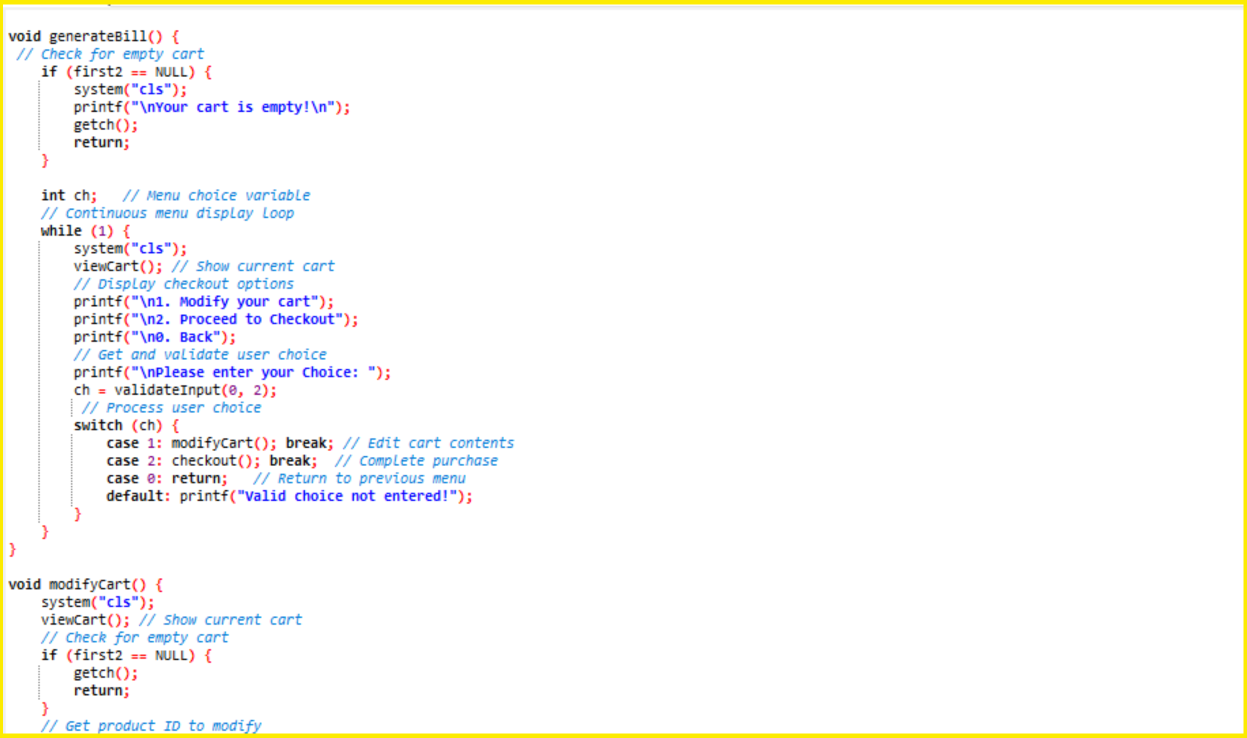
```c
    int id, qty;
    // Get product ID with validation
    printf("\nEnter the ID of the product you wish to add to cart: ");
    id = validateInput(1, 9999);
    // Check if product exists in inventory
    // Check if product exists
    int productExists = 0;
    for (ptr = first; ptr != NULL; ptr = ptr->next) {
        if (ptr->id == id) {
            productExists = 1;
            break;
        }
    }

    if (!productExists) {
        printf("\nProduct Not Found!!");
        getch();
        return;
    }
    // Get quantity with validation (1-10)
    printf("\nEnter the quantity[1-10]: ");
    qty = validateInput(1, 10);

    checkStock(id, qty);

    // Check if product already in cart
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        if (ptr2->id == id) {  // Update existing cart item quantity
            ptr2->qty += qty;
            // Enforce maximum quantity of 10
            if (ptr2->qty > 10) {
                ptr2->qty = 10;
                printf("\nQuantity limited to 10 as per policy.");
            }
            saveCartToFile();  // Save updated cart
            printf("\nProduct quantity updated in cart!");
            getch();
            return;
        }
    }
```



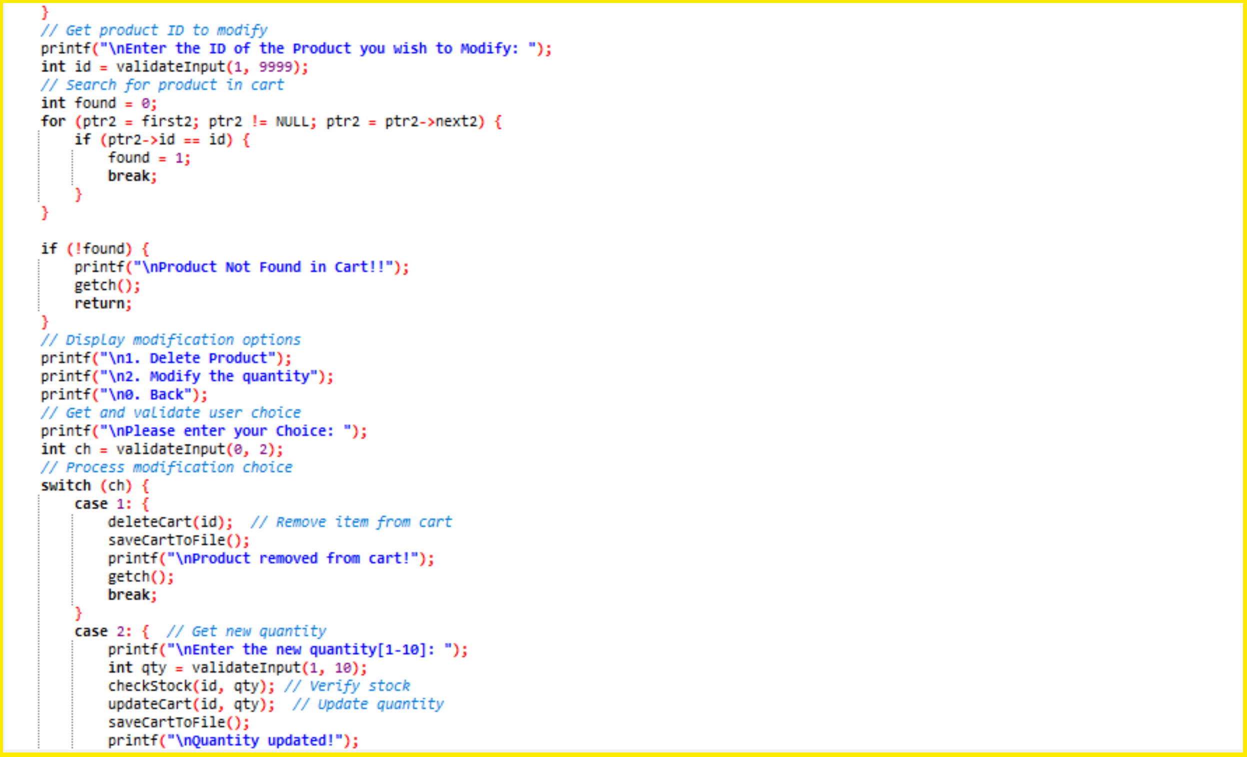
```c
        }
    }

    // Add new item to cart
    newnode2 = create_node2(id, qty);
    if (newnode2 == NULL) {
        printf("\nFailed to add to cart.");
        getch();
        return;
    }
    // Add to cart linked list
    if (first2 == NULL) {
        first2 = last2 = newnode2; // First item in cart
    } else {
        last2->next2 = newnode2;  // Append to existing cart
        last2 = newnode2;
    }

    saveCartToFile();  // Save cart state
    printf("\nProduct Added to Cart Successfully!!");
    // Ask if user wants to add more products
    printf("\nDo you want to add another product[Y/N]? ");
    char choice;
    scanf(" %c", &choice);
    if (toupper(choice) == 'Y') {
        addToCart();  // Recursively call to add another
    }
}

void checkStock(int id, int qty) {
    // Check all products for matching ID
    for (ptr = first; ptr != NULL; ptr = ptr->next) {
        if (ptr->id == id && ptr->qty < qty) {
            // Not enough stock available
            printf("\nNot enough stock available. Only %d left.", ptr->qty);
            printf("\nPress any key to continue...");
            getch();
            purchaseProduct();  // Return to purchase menu
            return;
        }
    }
```



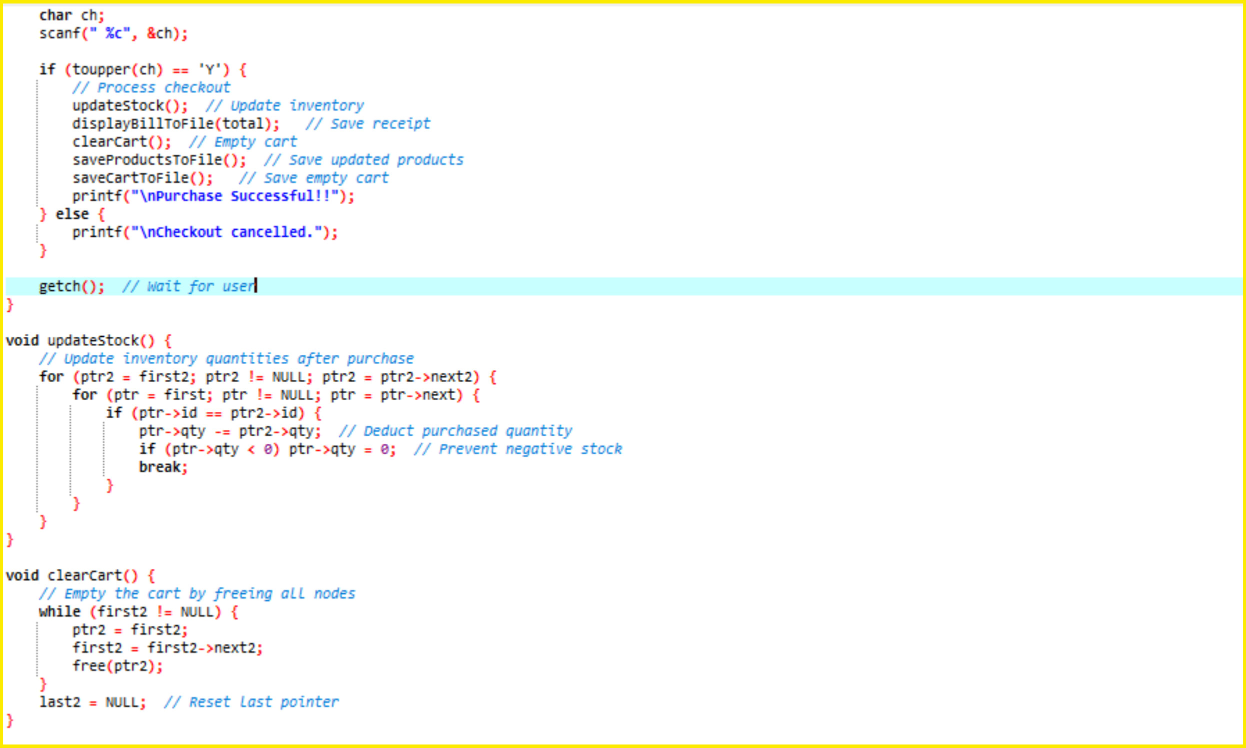
```c
void viewCart() {
    system("cls"); // Clear screen
    // Check for empty cart
    if (first2 == NULL) {
        ISEMPTY;  // Show empty message
        printf("No Products Available in Cart. \n");
        getch();
        return;
    }
    // Display cart header
    printf("=========================================================\n\n");
    printf("\t\t PRODUCTS IN CART\n\n");
    printf("=========================================================\n\n");
    printf("ID\tName\tQty\tPrice(Rs.)\tSubtotal\n\n");

    int total = 0;  // Initialize total amount
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        for (ptr = first; ptr != NULL; ptr = ptr->next) {
            if (ptr->id == ptr2->id) { // Calculate and display line item
                int subtotal = ptr2->qty * ptr->price;
                total += subtotal;
                printf("%d\t%s\t%d\t%d\t\t%d\n",
                      ptr->id, ptr->name, ptr2->qty, ptr->price, subtotal);
                break;
            }
        }
    }
    // Display cart footer with total
    printf("=========================================================\n");
    printf("TOTAL: Rs. %d\n", total);
    printf("=========================================================\n");

    printf("\nPress any key to continue...");
    getch(); // Wait for user
}

void generateBill() {
 // Check for empty cart
    if (first2 == NULL) {
        system("cls");
```



```c
void generateBill() {
 // Check for empty cart
    if (first2 == NULL) {
        system("cls");
        printf("\nYour cart is empty!\n");
        getch();
        return;
    }

    int ch;   // Menu choice variable
    // Continuous menu display loop
    while (1) {
        system("cls");
        viewCart(); // Show current cart
        // Display checkout options
        printf("\n1. Modify your cart");
        printf("\n2. Proceed to Checkout");
        printf("\n0. Back");
        // Get and validate user choice
        printf("\nPlease enter your Choice: ");
        ch = validateInput(0, 2);
         // Process user choice
        switch (ch) {
            case 1: modifyCart(); break; // Edit cart contents
            case 2: checkout(); break;  // Complete purchase
            case 0: return;   // Return to previous menu
            default: printf("Valid choice not entered!");
        }
    }
}

void modifyCart() {
    system("cls");
    viewCart(); // Show current cart
    // Check for empty cart
    if (first2 == NULL) {
        getch();
        return;
    }
    // Get product ID to modify
```

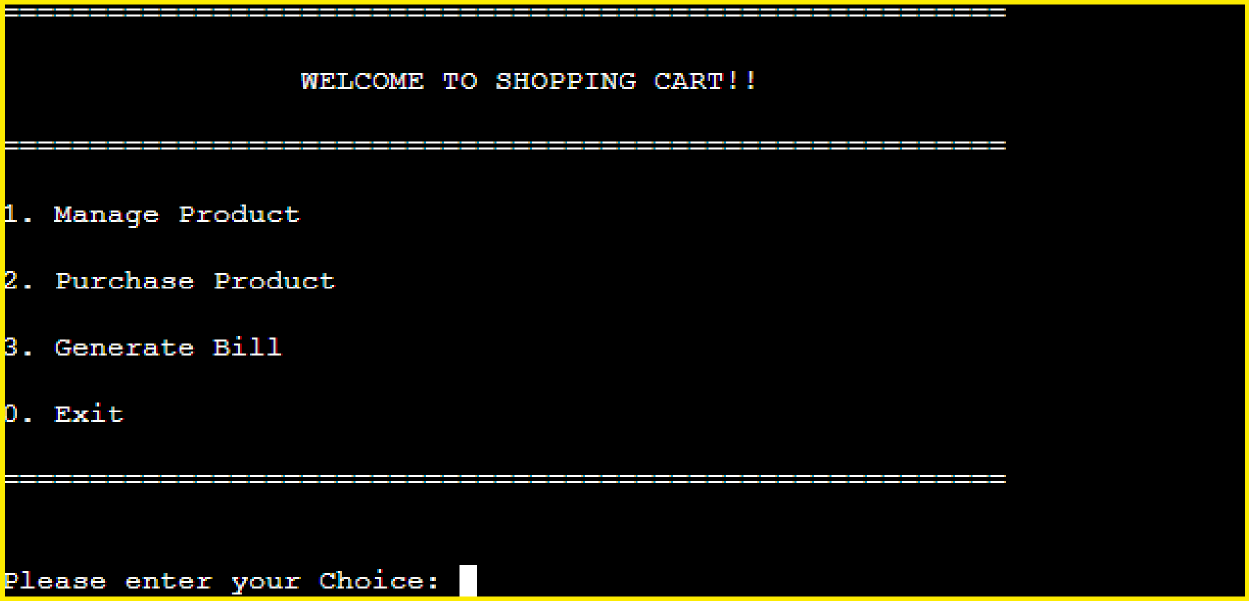


```c
    }
    // Get product ID to modify
    printf("\nEnter the ID of the Product you wish to Modify: ");
    int id = validateInput(1, 9999);
    // Search for product in cart
    int found = 0;
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        if (ptr2->id == id) {
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("\nProduct Not Found in Cart!!");
        getch();
        return;
    }
    // Display modification options
    printf("\n1. Delete Product");
    printf("\n2. Modify the quantity");
    printf("\n0. Back");
    // Get and validate user choice
    printf("\nPlease enter your Choice: ");
    int ch = validateInput(0, 2);
    // Process modification choice
    switch (ch) {
        case 1: {
            deleteCart(id);  // Remove item from cart
            saveCartToFile();
            printf("\nProduct removed from cart!");
            getch();
            break;
        }
        case 2: {  // Get new quantity
            printf("\nEnter the new quantity[1-10]: ");
            int qty = validateInput(1, 10);
            checkStock(id, qty); // Verify stock
            updateCart(id, qty);  // Update quantity
            saveCartToFile();
            printf("\nQuantity updated!");
```

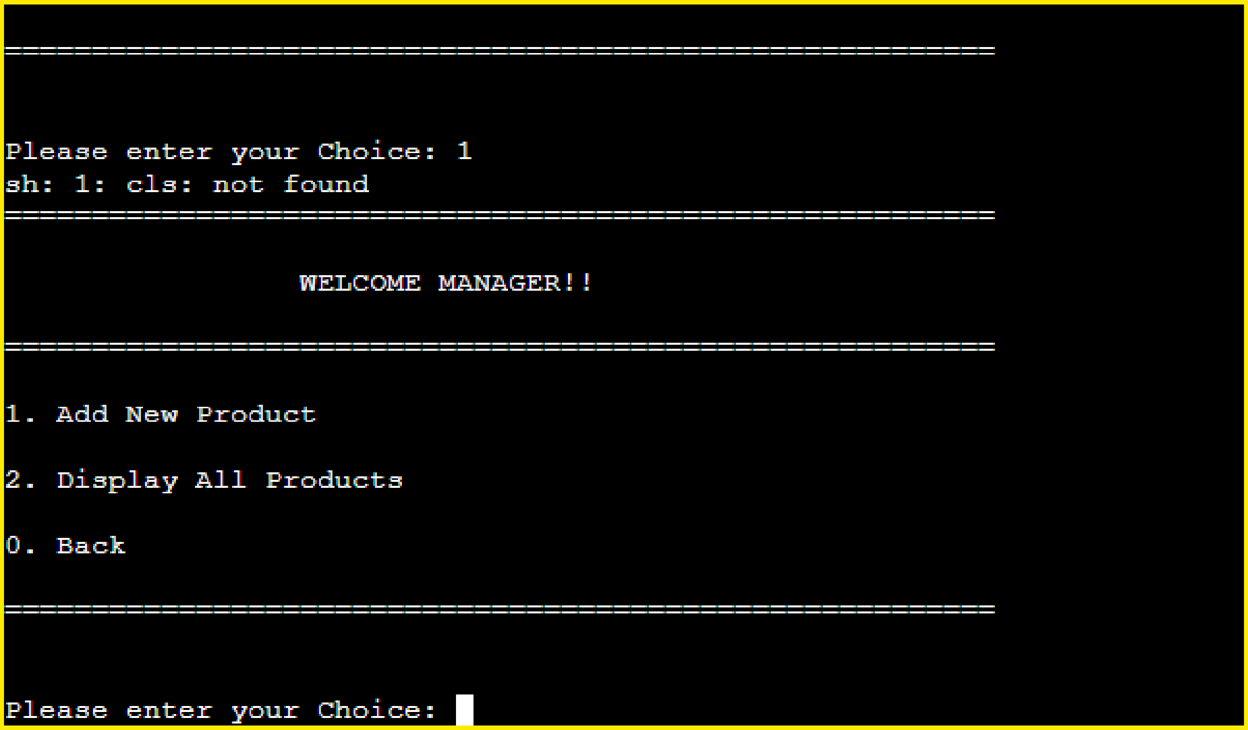


```
    char ch;
    scanf(" %c", &ch);

    if (toupper(ch) == 'Y') {
        // Process checkout
        updateStock();  // Update inventory
        displayBillToFile(total);   // Save receipt
        clearCart();  // Empty cart
        saveProductsToFile();  // Save updated products
        saveCartToFile();   // Save empty cart
        printf("\nPurchase Successful!!");
    } else {
        printf("\nCheckout cancelled.");
    }

    getch();  // Wait for user
}

void updateStock() {
    // Update inventory quantities after purchase
    for (ptr2 = first2; ptr2 != NULL; ptr2 = ptr2->next2) {
        for (ptr = first; ptr != NULL; ptr = ptr->next) {
            if (ptr->id == ptr2->id) {
                ptr->qty -= ptr2->qty;  // Deduct purchased quantity
                if (ptr->qty < 0) ptr->qty = 0;  // Prevent negative stock
                break;
            }
        }
    }
}

void clearCart() {
    // Empty the cart by freeing all nodes
    while (first2 != NULL) {
        ptr2 = first2;
        first2 = first2->next2;
        free(ptr2);
    }
    last2 = NULL;  // Reset last pointer
}
```

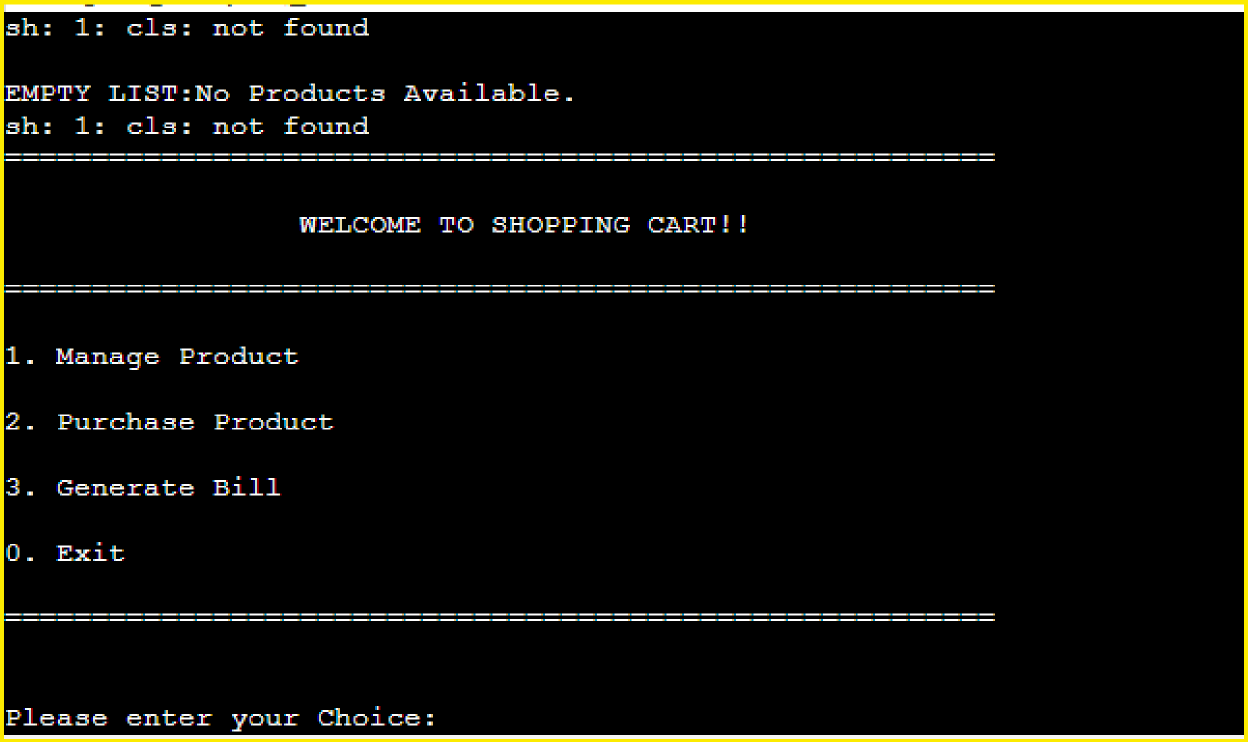
Sample Output:



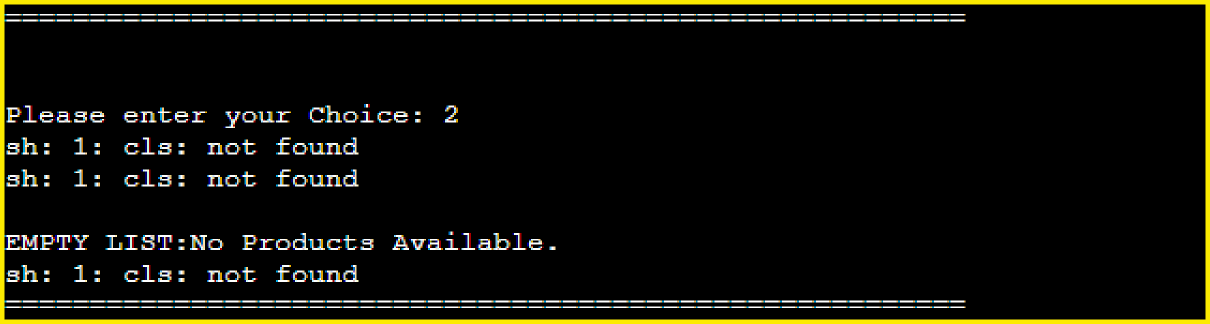
```
============================================================

                  WELCOME TO SHOPPING CART!!

============================================================

1. Manage Product

2. Purchase Product

3. Generate Bill

0. Exit

============================================================


Please enter your Choice:
```



```
=========================================================


Please enter your Choice: 1
sh: 1: cls: not found
=========================================================

                 WELCOME MANAGER!!

=========================================================

1. Add New Product

2. Display All Products

0. Back

=========================================================


Please enter your Choice:
```



```
sh: 1: cls: not found

EMPTY LIST:No Products Available.
sh: 1: cls: not found
=========================================================

                 WELCOME TO SHOPPING CART!!

=========================================================

1. Manage Product

2. Purchase Product

3. Generate Bill

0. Exit

=========================================================


Please enter your Choice:
```



```
================================================================

Please enter your Choice: 2
sh: 1: cls: not found
sh: 1: cls: not found

EMPTY LIST:No Products Available.
sh: 1: cls: not found
================================================================
```

Testing:

Test No.	Feature Tested	Input	Expected Output	Result
TC1	Add Product	ID: 101, Name: Apple, Price: 50, Qty: 20	Product Added Successfully	✅
TC2	View All Products	-	Displays product list	✅
TC3	Add to Cart	ID: 101, Qty: 2	Product Added to Cart Successfully	✅
TC4	View Cart	-	Shows cart items with subtotal	✅
TC5	Modify Cart	Change quantity to 4	Quantity updated	✅
TC6	Generate Bill & Checkout	Confirm checkout (`Y`)	Receipt saved to `receipt.txt`, stock updated	✅
TC7	Delete Product	ID: 101 (not in cart)	Product deleted successfully	✅
TC8	File Save/Load	Restart program	Previous products and cart restored	✅

Conclusion

The project successfully demonstrates a console-based Online Shopping Cart System implemented in C using linked lists and file handling. Users can:

- Manage products (add, view, and delete)
- Add products to the cart
- View or modify the shopping cart
- Generate a bill and checkout
- Store and load data persistently using binary files (products.txt and cart.dat)
- Generate a text receipt (receipt.txt)

It is a practical application of:

- Data structures (linked lists)
- Input validation
- Dynamic memory allocation
- Persistent storage with file I/O