**ONLINE FOOD ORDERING SYSTEM**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**A. YOGYA SAI SRI -24KB1A0504**

**A. MANEESHA -24KB1A0513**

**B. SHAMITHA -24KB1A0542**

**B. ROOPA -24KB1A0572**

**Under the Guidance of**

**MRS.SRUTHI**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY**

**N.B.K.R. INSTITUTE OF SCIENCE &TECHNOLOGY**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



###### CERTIFICATE

This is to certify that the project report entitled YOUR PROJECT TITLE being submitted by

**A. YOGYA SAI SRI -24KB1A0504**

**A. MANEESHA -24KB1A0513**

**B. SHAMITHA -24KB1A0542**

**B. ROOPA -24KB1A0572**

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to THE N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY is a record of bonafied work carried out under my guidance and supervision.

|  |  |
| --- | --- |
| **MRS.SRUTHI**  **Designation** | **Dr. HOD Name**  **M.Tech, Ph.D**  **Head of the Department** |

**DECLARATION**

I hereby declare that the dissertation entitled **ONLINE FOOD ORDERING SYSTEM** submitted for the B. Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

**A. YOGYA SAI SRI-24KB1A0504**

**A. MANEESHA -24KB1A0513**

**B. SHAMITHA -24KB1A0542**

**B. ROOPA -24KB1A0572**

Place: Vidyanagar

Date: 05-05-2025

### 

First and foremost, I extend my heartfelt thanks to my MRS. SRUTHI, whose expert guidance, encouragement, and insightful feedback played a crucial role

### Acknowledgement :-

I would like to express my sincere gratitude to all those who have supported and guided me throughout the development of the **Online Food Ordering System**

shaping this project. Their continuous support kept me focused and motivated throughout the development process.

I am also grateful to my institution N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY for providing me with the opportunity and necessary resources to carry out this project.

Special thanks to my classmates, friends, and family for their constant support, valuable suggestions, and encouragement during the entire course of this work.

Lastly, I appreciate all the developers and online communities whose tutorials, forums, and shared resources helped me overcome technical challenges during the creation of this system.

This project has been a valuable learning experience, and I am thankful to everyone who contributed to its successful completion.

**Abstract**

This project showcases the development of an **Online Food Ordering System** implemented in the **C programming language**, utilizing core **data structures** to manage operations efficiently. The system allows users to view a food menu, place orders, and review their current and past selections through a text-based interface. A static array is used to store the food menu, enabling quick and direct access to available items. To manage user orders dynamically, **singly linked lists** are employed, allowing flexible and efficient addition or removal of orders without memory wastage. The design emphasizes modular programming, user interaction, and the practical application of data structures to simulate real-world systems. This project demonstrates how foundational concepts like arrays and linked lists can be integrated to build a functional, menu-driven food ordering application.

**TABLE OF CONTENT :-**

1. **Introduction  
   1.1 Overview  
   1.2 Objective  
   1.3 Scope of the Project**
2. **System Requirements  
   2.1 Hardware Requirements  
   2.2 Software Requirements**
3. **Literature Review / Existing Systems  
   3.1 Traditional Food Ordering Methods  
   3.2 Existing Online Systems and Their Limitations**
4. **System Design  
   4.1 System Architecture  
   4.2 Data Flow Diagram (DFD)  
   4.3 Use Case Diagram (if applicable)**
5. **Implementation Details  
   5.1 Programming Language Used  
   5.2 Data Structures Used**
   * **5.2.1 Array (for Menu)**
   * **5.2.2 Linked List (for Orders)  
     5.3 Key Modules and Functions**
   * **5.3.1 Display Menu**
   * **5.3.2 Place Order**
   * **5.3.3 View Order**
   * **5.3.4 Cancel Order**

**6.conclusion**

**Chapter 1**

**1.Introduction:-**

**It allows users to view categorized menus and place food orders by selecting items and quantities.  
The system stores each order in a linked list and generates a final bill upon completion.**

**1.1Overview :-  
The system displays menus across categories such as Starters, Soups, Juices, Biriyanis, Curries, and Ice Creams.  
Users can navigate through categories, select food items, and track orders dynamically.It uses arrays for item storage and linked lists for managing ordered items efficiently.**

**1.2Objective :-  
The main objective is to simulate a basic digital food ordering experience in a restaurant.  
It aims to teach dynamic memory allocation, structured programming, and menu-driven design.  
Another goal is to ensure accurate billing and enhance user interaction in a simple CLI format.**

**1.3Scope :-  
This system is suitable for small-scale restaurant setups or academic demonstration purposes.  
It can be extended with features like order cancellation, file storage, or graphical interface.  
Overall, it provides a modular foundation for more complex restaurant management software.**

**Chapter 2**

**2.1🖥️ Hardware Requirements:**

1. **Processor: Intel Pentium IV or higher**
2. **RAM: Minimum 512 MB (1 GB or more recommended)**
3. **Hard Disk: At least 100 MB of free space**
4. **Display: Standard monitor (console output only)**
5. **Input Devices: Keyboard (for menu navigation and data input)**

**2.2💽 Software Requirements:**

1. **Operating System: Windows, Linux, or macOS**
2. **Compiler: GCC (for Linux/macOS) or MinGW (for Windows)**
3. **IDE (Optional):**
   * **Code:Blocks**
   * **Dev-C++**
   * **Visual Studio Code with C/C++ extension**
4. **Language: C (ANSI C standard)**
5. **Other Tools: Command-line terminal or shell for execution**

**Chapter 3**

**3.Literature Review**

**3.1 Traditional Food Ordering Method :-  
Traditionally, food ordering in restaurants has been a manual process. Customers physically visit the restaurant, browse a printed menu, and place their orders through a waiter or cashier. This method often involves verbal communication, handwritten bills, and manual record-keeping. While simple, this approach is prone to human errors, order mix-ups, and inefficiencies during peak hours. Additionally, it offers no real-time order tracking or digital record maintenance.**

**3.2 Existing Online Food Ordering Systems :-  
In recent years, online food ordering platforms like Swiggy, Zomato, and Uber Eats have gained popularity. These systems allow users to browse digital menus, place orders, make payments online, and get food delivered to their location. Restaurants also implement POS (Point of Sale) systems with touchscreen interfaces for staff to manage orders digitally. Some advanced systems integrate with inventory management, CRM, and analytics tools for business optimization.**

**3.3 Limitations of Current Systems :-  
Despite their convenience, existing online systems have several drawbacks. High commission fees imposed by third-party apps reduce restaurant profits. Technical glitches or poor internet connectivity can disrupt operations. Small restaurants may find these platforms expensive or complex to integrate. Moreover, customer data is often controlled by the third-party platforms, limiting direct engagement between restaurants and their customers.**

**Chapter 4**

**4. System Design**

**4.1 System Architecture**

**Architecture Type: Monolithic Console-Based Application (Single-tier)**

1. **📌 Components:**
2. **User Interface Layer (via printf, scanf)**
   * **Handles user input and displays menus/orders.**
3. **Business Logic Layer**
   * **Manages item selection, quantity validation, price calculation, and bill generation.**
4. **Data Layer**
   * **Stores menu items as arrays.**
   * **Manages dynamic order list via a linked list (struct order).**

**4.2 Data Flow Diagram (DFD)**

**Level 0 – Context Diagram:**

**+-----------------+ +-----------------------+**

**| Customer |<-------------> | Restaurant System |**

**+-----------------+ +-----------------------+**

**| 1. View Menu**

**| 2. Place Order**

**| 3. View Bill**

**Level 1 – Detailed Flow:**

**[Customer]**

**|**

**v**

**[Menu Interface] -----> [Menu Data (Arrays)]**

**4.3 Use Case Diagram (if applicable)**

**Actors:**

* **Customer (Interacts through console)**

**Use Cases:**

* **View Menu**
* **Select Category**
* **Choose Item**
* **Specify Quantity**
* **Place Order**
* **View Final Bill**

**+----------------+**

**| Customer |**

**+----------------+**

**|**

**+----------+----------+**

**| | |**

**+----------+ +----------+ +----------+**

**| View | | Place | | View |**

**| Menu | | Order | | Bill |**

**+----------+ +----------+ +----------+**

**v**

**[Order Handler] ------> [Order List (Linked List)]**

**|**

**v**

**[Bill Generator] ------> [Display Bill]**

**Chapter 5**

**5. Implementation Details**

**5.1 Programming Language Used**

* **C Programming Language**
  + **Chosen for its efficiency and low-level memory management using pointers.**
  + **Ideal for command-line based applications where speed and control over memory (like linked lists) are important.**

**5.2 Data Structures Used**

* **5.2.1 Array (for Menu)**
* **Used to store:**
  + **Menu item names (char\* itemName[])**
  + **Corresponding prices (float itemPrices[])**
* **Arrays make it easy to loop through and display items category-wise.**

**Example:**

**char \*StarterItems[] = {"Egg Roll", "Veg Spring Roll", ...};**

**float starterPrices[] = {30.0, 40.0, ...};**

* **5.2.2 Linked List (for Orders)**
* **Struct-based singly linked list for managing dynamic order entries.**
* **Each node represents an item ordered, including:**
  + **Item name**
  + **Quantity**
  + **Total price for that item**
* **Allows flexibility in adding any number of items at runtime.**
* **Structure:**

**struct order {**

**char itemName[100];**

**int qty;**

**float price;**

**struct order \*next; };**

* 1. **Key Modules and Functions**

**5.3.1 displayMenu(...)**

* **Purpose: Show menu items of a selected category.**
* **Inputs: Array of item names, prices, item count, and title.**
* **Output: Formatted menu list printed to the console.**
* **Example Call:**
* **displayMenu(StarterItems, starterPrices, starterCount, "STARTERS");**
* **5.3.2 takeOrder(...)**
* **Purpose: Add an order to the linked list based on user input.**
* **Process:**
  + **Get item number and quantity from user.**
  + **Validate input.**
  + **Allocate memory dynamically and insert into linked list.**
* **Handles: New order creation and appending to order list.**
* **5.3.3 displayBill(...)**
* **Purpose: Print all ordered items and the total amount.**
* **Iterates through the linked list and prints:**
  + **Item name**
  + **Quantity**
  + **Total item price**
* **Also displays: Final total amount to pay.**
* **5.3.4 cancelOrder() *(To Be Implemented)***
* **Suggested Addition:**
  + **Allow user to remove an item from the order list.**
  + **Would require:**
    - **Input: Item name or number.**
    - **Traversal of the linked list.**
    - **Removal and freeing of selected node.**

**Prototype Suggestion:**

**void cancelOrder(struct order \*\*head, const char \*itemName);**

**Source code :-**

**#include <stdio .h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct order {**

**char itemName[100];**

**int qty;**

**float price;**

**struct order \*next;**

**};**

**char \*StarterItems[] = {"Egg Roll", "Veg Spring Roll", "Chicken Roll", "French Fries", "Momos"};**

**float starterPrices[] = {30.0, 40.0, 50.0, 40.0, 60.0};**

**char \*vegSoupItems[] = {"Veg Clear Soup", "Veg Corn Soup", "Cream of Tomato", "Veg Lemon Coriander"};**

**float vegSoupPrices[] = {149.0, 149.0, 149.0, 149.0};**

**char \*nonVegSoupItems[] = {"Chicken Clear", "Chicken Corn", "Chicken Dragon", "Chicken Manchow"};**

**float nonVegSoupPrices[] = {169.0, 169.0, 169.0, 169.0};**

**char \*fruiteJucieItems[] = {"Mango Juice", "Grape Juice", "Lemonade", "Mojito", "Fresh Fruit Juice"};**

**float fruiteJuciePrices[] = {90.0, 90.0, 97.0, 177.0, 157.0};**

**char \*iceCreamItems[] = {"Chocolate", "Vanilla", "Black Currant", "Pista", "Caramel"};**

**float iceCreamPrices[] = {100.0, 90.0, 150.0, 130.0, 150.0};**

**char \*vegBiriyaniItems[] = {"Veg Biriyani", "Paneer Biriyani", "Mushroom Biriyani", "Meal Maker", "Kaju Biriyani"};**

**float vegBiriyaniPrices[] = {100.0, 120.0, 150.0, 90.0, 140.0};**

**char \*nonVegBiriyaniItems[] = {"Chicken Biriyani", "Mutton Biriyani", "Fish Biriyani", "Egg Biriyani", "Prawns Biriyani"};**

**float nonVegBiriyaniPrices[] = {150.0, 200.0, 180.0, 100.0, 160.0};**

**char \*vegCurriesItems[] = {"Paneer Butter", "Palakura Pappu", "Mix Veg", "Guthi Vankaya", "Tomato Pappu"};**

**float vegCurriesPrices[] = {90.0, 40.0, 40.0, 40.0, 40.0};**

**char \*nonVegCurriesItems[] = {"Chicken Tikka", "Kadai Chicken", "Mutton Rogan", "Mughlai Chicken", "Chicken Kandhari"};**

**float nonVegCurriesPrices[] = {349.0, 349.0, 369.0, 349.0, 369.0};**

**// Function to display a menu**

**void displayMenu(char \*items[], float prices[], int count, const char \*title) {**

**printf("\n--- %s ---\n", title);**

**for (int i = 0; i < count; i++) {**

**printf("%d. %s - ₹%.2f\n", i + 1, items[i], prices[i]);**

**}**

**}**

**void takeOrder(char \*items[], float prices[], int count, struct order \*\*head) {**

**int itemNo, qty;**

**printf("Enter item number to order (1 to %d): ", count);**

**scanf("%d", &itemNo);**

**if (itemNo < 1 || itemNo > count) {**

**printf("Invalid item number!\n");**

**return;**

**}**

**printf("Enter quantity: ");**

**scanf("%d", &qty);**

**if (qty <= 0) {**

**printf("Invalid quantity!\n");**

**return;**

**}**

**struct order \*newOrder = (struct order \*)malloc(sizeof(struct order));**

**strcpy(newOrder->itemName, items[itemNo - 1]);**

**newOrder->qty = qty;**

**newOrder->price = prices[itemNo - 1] \* qty;**

**newOrder->next = NULL;**

**if (\*head == NULL) {**

**\*head = newOrder;**

**} else {**

**struct order \*temp = \*head;**

**while (temp->next != NULL)**

**temp = temp->next;**

**temp->next = newOrder;**

**}**

**printf("Item added to order!\n");**

**}**

**void displayBill(struct order \*head) {**

**float total = 0;**

**printf("\n--- BILL ---\n");**

**printf("%-25s %-10s %-10s\n", "Item", "Qty", "Price");**

**while (head != NULL) {**

**printf("%-25s %-10d ₹%-9.2f\n", head->itemName, head->qty, head->price);**

**total += head->price;**

**head = head->next;**

**}**

**printf("-------------------------------\n");**

**printf("Total Amount: ₹%.2f\n", total);**

**}**

**int main() {**

**int choice;**

**struct order \*orderItemsList = NULL;**

**// Count array elements**

**int starterCount = sizeof(starterPrices) / sizeof(starterPrices[0]);**

**int vegSoupCount = sizeof(vegSoupPrices) / sizeof(vegSoupPrices[0]);**

**int nonVegSoupCount = sizeof(nonVegSoupPrices) / sizeof(nonVegSoupPrices[0]);**

**int fruiteJucieCount = sizeof(fruiteJuciePrices) / sizeof(fruiteJuciePrices[0]);**

**int icecreamCount = sizeof(iceCreamPrices) / sizeof(iceCreamPrices[0]);**

**int vegBiriyaniCount = sizeof(vegBiriyaniPrices) / sizeof(vegBiriyaniPrices[0]);**

**int nonVegBiriyaniCount = sizeof(nonVegBiriyaniPrices) / sizeof(nonVegBiriyaniPrices[0]);**

**int vegCurriesCount = sizeof(vegCurriesPrices) / sizeof(vegCurriesPrices[0]);**

**int nonVegCurriesCount = sizeof(nonVegCurriesPrices) / sizeof(nonVegCurriesPrices[0]);**

**displayMenu(StarterItems, starterPrices, starterCount, "STARTERS");**

**displayMenu(vegSoupItems, vegSoupPrices, vegSoupCount, "VEG SOUPS");**

**displayMenu(nonVegSoupItems, nonVegSoupPrices, nonVegSoupCount, "NON VEG SOUPS");**

**displayMenu(fruiteJucieItems, fruiteJuciePrices, fruiteJucieCount, "FRUITE JUICES");**

**displayMenu(iceCreamItems, iceCreamPrices, icecreamCount, "ICE CREAMS");**

**displayMenu(vegBiriyaniItems, vegBiriyaniPrices, vegBiriyaniCount, "VEG BIRIYANI");**

**displayMenu(nonVegBiriyaniItems, nonVegBiriyaniPrices, nonVegBiriyaniCount, "NON VEG BIRIYANI");**

**displayMenu(vegCurriesItems, vegCurriesPrices, vegCurriesCount, "VEG CURRIES");**

**displayMenu(nonVegCurriesItems, nonVegCurriesPrices, nonVegCurriesCount, "NON VEG CURRIES");**

**do {**

**printf("\nChoose category to order:\n");**

**printf("1. Starters\n2. Veg Soup\n3. Non-Veg Soup\n4. Fruite Juice\n5. Ice Creams\n6. Veg Biriyani\n7. Non-Veg Biriyani\n8. Veg Curries\n9. Non-Veg Curries\n0. Finish Order\nEnter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**displayMenu(StarterItems, starterPrices, starterCount, "STARTERS");**

**takeOrder(StarterItems, starterPrices, starterCount, &orderItemsList);**

**break;**

**case 2:**

**displayMenu(vegSoupItems, vegSoupPrices, vegSoupCount, "VEG SOUPS");**

**takeOrder(vegSoupItems, vegSoupPrices, vegSoupCount, &orderItemsList);**

**break;**

**case 3:**

**displayMenu(nonVegSoupItems, nonVegSoupPrices, nonVegSoupCount, "NON VEG SOUPS");**

**takeOrder(nonVegSoupItems, nonVegSoupPrices, nonVegSoupCount, &orderItemsList);**

**break;**

**case 4:**

**displayMenu(fruiteJucieItems, fruiteJuciePrices, fruiteJucieCount, "FRUITE JUICES");**

**takeOrder(fruiteJucieItems, fruiteJuciePrices, fruiteJucieCount, &orderItemsList);**

**break;**

**case 5:**

**displayMenu(iceCreamItems, iceCreamPrices, icecreamCount, "ICE CREAMS");**

**takeOrder(iceCreamItems, iceCreamPrices, icecreamCount, &orderItemsList);**

**break;**

**case 6:**

**displayMenu(vegBiriyaniItems, vegBiriyaniPrices, vegBiriyaniCount, "VEG BIRIYANI");**

**takeOrder(vegBiriyaniItems, vegBiriyaniPrices, vegBiriyaniCount, &orderItemsList);**

**break;**

**case 7:**

**displayMenu(nonVegBiriyaniItems, nonVegBiriyaniPrices, nonVegBiriyaniCount, "NON VEG BIRIYANI");**

**takeOrder(nonVegBiriyaniItems, nonVegBiriyaniPrices, nonVegBiriyaniCount, &orderItemsList);**

**break;**

**case 8:**

**displayMenu(vegCurriesItems, vegCurriesPrices, vegCurriesCount, "VEG CURRIES");**

**takeOrder(vegCurriesItems, vegCurriesPrices, vegCurriesCount, &orderItemsList);**

**break;**

**case 9: displayMenu(nonVegCurriesItems, nonVegCurriesPrices, nonVegCurriesCount, "NON VEG CURRIES");**

**takeOrder(nonVegCurriesItems, nonVegCurriesPrices, nonVegCurriesCount, &orderItemsList);**

**break;**

**case 0:**

**printf("Finalizing order...\n");**

**break;**

**default:**

**printf("Invalid choice!\n");**

**}**

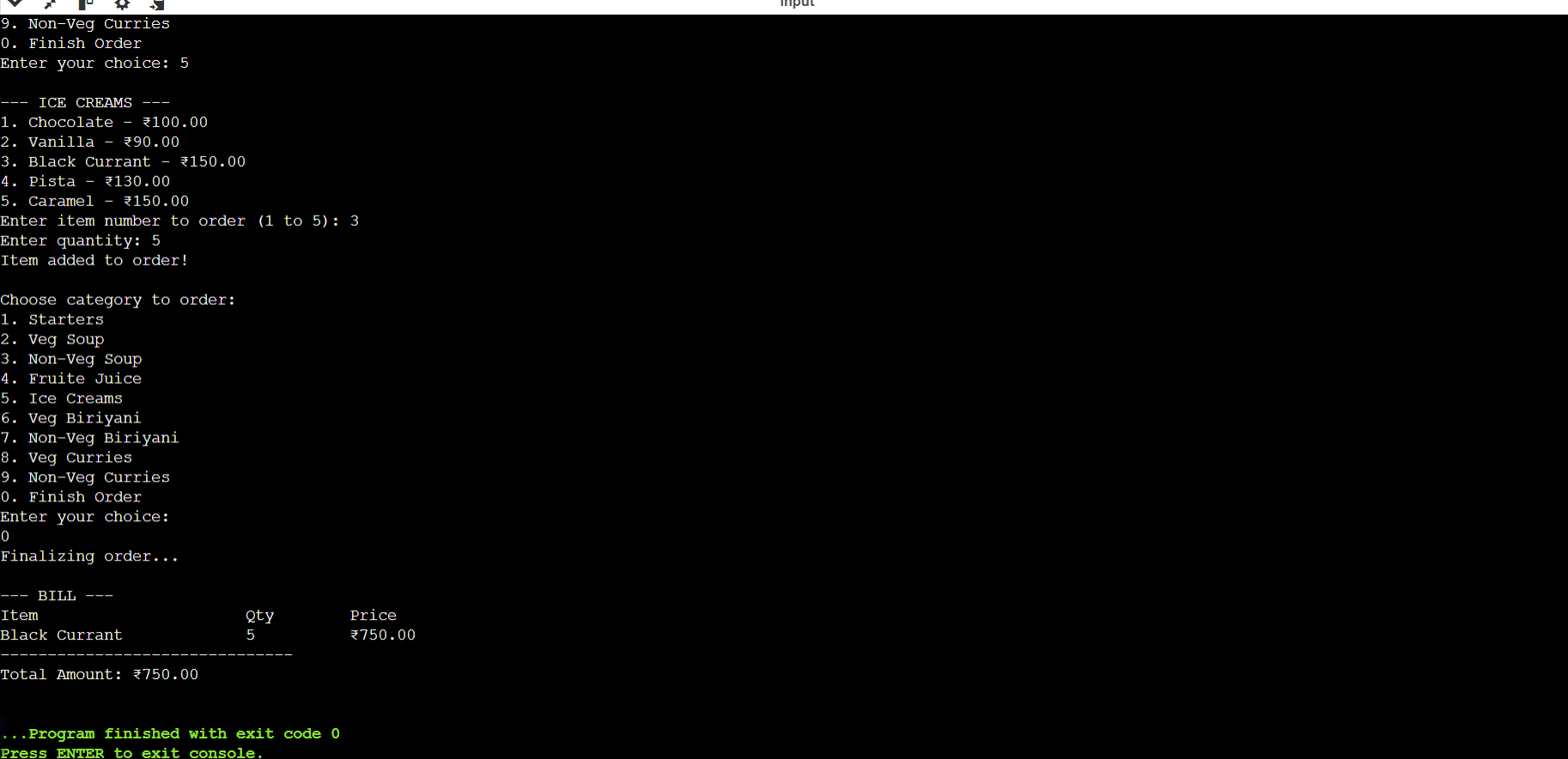
**} while (choice != 0);**

**displayBill(orderItemsList);**

**return 0;**

**}**

# OUTPUT:-

****

**Chapter 6**

**6.Conclusion :-**

**The online food ordering system designed in C effectively demonstrates the use of core programming concepts such as arrays, structures, pointers, and linked lists. It provides a user-friendly interface for browsing a categorized menu, placing multiple orders, and generating a detailed bill dynamically. The system ensures modularity through well-defined functions like displayMenu, takeOrder, and displayBill, making it easy to maintain and extend.**

**While currently focused on a console-based application, this system lays a solid foundation for future enhancements, such as:**

* **Order cancellation and modification features**
* **File storage for order history**
* **GUI integration for real-world deployment**
* **Tax, discount, and payment gateway simulation**

**Overall, the system demonstrates how foundational C programming can be leveraged to create practical, real-world applications with structured data flow and dynamic memory handling.**