# 🧾 Supermarket Billing System Using Stack in C

**📄 Page 1: Title Page**

**Project Title**: Supermarket Billing System Using Stack in C  
**Submitted By**: Danish Gupta, Aditya Rangta   
**Roll Number**: 24bca10138, 24bca10156  
**Course**: BCA  
**Semester**: 2nd SEM   
**Department**: UIC  
**College**: CHANDIGARH UNVERSITY  
**Date of Submission**: 21/04/2025

# 📄 Page 2: Introduction

**Introduction**

In today’s fast-paced retail environment, supermarkets rely on efficient billing systems to manage their operations. A billing system is an essential part of a point-of-sale system that helps manage transactions and customer purchases. This project, **"Supermarket Billing System Using Stack in C"**, is a console-based application designed to replicate the basic functionality of a billing system.

The uniqueness of this system lies in its use of the **stack data structure**, following the **Last-In-First-Out (LIFO)** principle. This concept is particularly useful when implementing a feature like **canceling the last added item**, which is commonly found in supermarket billing systems. The application allows users to add items to the bill, remove the last item added, display the current bill items, and generate the final bill.

This project provides a practical implementation of stack operations in a real-world scenario and enhances understanding of how data structures can be used effectively to manage data flow and system features.

# 📄 Page 3: Objectives

**Objectives of the Project**

The main goal of this project is to design and implement a simplified supermarket billing system using the stack data structure in the C programming language. The specific objectives are:

1. **Understand Stack Operations**: Learn how stack operations such as push, pop, and peek can be applied in a billing system.
2. **Simulate Billing Process**: Develop a simple interface that simulates the process of adding, removing, and displaying items in a bill.
3. **User Interaction**: Allow users to interact with the system through a menu-based interface to perform tasks such as:
   * Adding an item (name, price, quantity)
   * Removing the most recent item
   * Viewing all items in the current bill
   * Generating the final invoice with total amount
4. **Error Handling**: Implement proper error checks such as stack overflow (maximum item limit) and stack underflow (no items to remove).
5. **Educational Purpose**: Serve as a foundational learning tool for data structures, file handling, and C programming techniques in an applied scenario.

By achieving these objectives, students and developers will gain a better grasp of how fundamental data structures like stacks can be practically applied beyond theoretical exercises.

# 📄 Page 4: Tools and Technologies

**Tools and Technologies Used**

This project was implemented using the C programming language due to its simplicity and close-to-hardware execution speed. The key tools and technologies used in this project are:

**1. Programming Language: C**

* C is a powerful general-purpose language.
* Offers low-level memory access and is widely used in system software development.
* Suitable for implementing stack operations manually using arrays.

**2. Compiler: GCC / Code::Blocks / Turbo C**

* GCC (GNU Compiler Collection) is an industry-standard C compiler.
* Code::Blocks is an IDE that supports debugging and compiling with ease.
* Turbo C can be used in educational environments.

**3. Data Structures Used: Stack**

* Implemented using a structure with a fixed-size array.
* Supports LIFO operations required to manage the addition and cancellation of items in a bill.

**4. Operating System: Windows / Linux**

* The system is compatible with any OS that supports a C compiler.

**5. Optional Enhancements:**

* File handling for saving bills (not implemented in this version).
* GUI design for better user experience (can be done using C++ or Python in future versions).

# 📄 Page 5: System Design

**System Design and Modules**

The system consists of the following main modules:

**1. Add Item Module**

* Accepts item name, price, and quantity.
* Pushes the item to the top of the stack.
* Ensures that the stack is not full before adding.

**2. Remove Item Module**

* Removes the last item added (LIFO).
* Displays a confirmation message for the removed item.
* Ensures that the stack is not empty before popping.

**3. Display Current Bill**

* Traverses the stack from bottom to top.
* Displays each item with calculated total price (price × quantity).
* Provides a snapshot of the current bill status.

**4. Generate Final Bill**

* Traverses all stack items.
* Calculates and displays the grand total.
* Acts as the final checkout step.

**5. Exit Module**

* Terminates the program safely.
* Frees memory if needed (not applicable for static arrays).

The flow of operations is governed by a menu-driven loop in the main function. Users can choose from the available operations by entering the corresponding number.

# 📄 Page 6: Sample Code and Output

**Sample Code Snippet (Main Function Loop)**

c

CopyEdit

while (1) {

printf("1. Add Item\n2. Remove Last Item\n3. Display Bill\n4. Final Bill\n5. Exit\n");

scanf("%d", &choice);

switch (choice) {

case 1: /\* Add Item Code \*/ break;

case 2: /\* Pop Last Item \*/ break;

case 3: /\* Display Items \*/ break;

case 4: /\* Final Bill \*/ break;

case 5: exit(0);

}

}

**Sample Output**

mathematica

CopyEdit

==== Supermarket Billing Menu ====

1. Add Item

2. Remove Last Item

3. Display Current Bill

4. Generate Final Bill

5. Exit

Enter choice: 1

Enter item name: Milk

Enter price: 30

Enter quantity: 2

Item added: Milk | Price: 30.00 | Quantity: 2

Enter choice: 3

--- Current Bill Items ---

Item Price Qty Total

Milk 30.00 2 60.00

# 📄 Page 7: Conclusion

**Conclusion**

The Supermarket Billing System project demonstrates how foundational data structures can be practically used to build real-world applications. By leveraging the **stack data structure**, we effectively simulate features such as canceling the last item added—a common need in billing counters at retail stores.

This project reinforces the understanding of stack concepts in C while also emphasizing good programming practices like modularity, error handling, and user interaction. Through this project, we achieve a working model of a supermarket billing interface, and it serves as a base to build more complex systems such as:

* GST calculations
* Barcode scanning
* Inventory management
* User accounts and transaction history

Overall, the project provides an excellent opportunity to apply theoretical knowledge to solve practical problems using C and data structures.

# 📄 Page 8: Future Scope

**Future Enhancements**

The current system is designed for educational purposes and uses static memory for stack implementation. It can be significantly enhanced with the following features:

**1. File Handling**

* Save and retrieve bill history using .txt or .csv files.
* Helpful for generating daily, weekly, or monthly reports.

**2. Dynamic Stack (Linked List)**

* Replace static array with a linked list to remove size limitation.

**3. Graphical User Interface**

* Create a GUI using libraries like GTK (in C) or move to C++/Java/Python.

**4. Inventory Management**

* Add backend inventory to check availability.
* Reduce item quantity upon billing.

**5. Barcode Scanning**

* Integrate barcode reader for product code entry.

**6. Database Integration**

* Connect with a database like SQLite or MySQL to manage items.

These enhancements can make the system more robust and applicable in real-world retail environments.

***OUTPUT IMAGE:***

