**National University of Computer & Emerging Sciences, Karachi Fall-2025 School of Computing (BSCS, BSSE, BSCY, BSAI)**
**Assignment # 2**

| | |
|---|---|
| **Subject: Programming Fundamentals CS-1002** | **Post Date:** 21/10/2025 |
| **Total Marks: 50** | **Due Date:** 30/10/2025 |
| **Course Instructors: Javeria Farooq, Sobia Iftikhar, Abeeha Sattar, Saif ur Rehman, Syed Ahmed, Faisal Ali, Fizza Mansoor, Talha Shahid, Abuzar Zafar, Uzma Raza, Hajra Ahmed** | |

**Instructions**

- All Programs should be submitted in softcopy and should have .c file
- It should be obvious that submitting your work after the due date will result in zero points being awarded.
- Plagiarism (copying/cheating) and late submissions result in a zero mark.

---

**Question No. 1**

Liberty Books, a major bookstore chain in Pakistan, needs a new command-line application for its smaller branches to manage inventory. You must build this system using parallel arrays to track all book data, as it needs to be simple and efficient.

Core Requirements:

1. **Data Storage**: Use separate, parallel arrays to store information for up to 100 books:
   - int isbns[100];
   - char titles[100][50];
   - float prices[100];
   - int quantities[100]; A single index i will correspond to the same book across all
   - arrays (e.g., isbns[i] and prices[i] refer to the same book).
2. **Main Menu**: The program must be menu-driven using a switch statement, offering options to:
   - **Add New Book**: Add a book's details into the arrays. Prevent duplicate ISBNs.
   - **Process a Sale**: Ask for an ISBN and the number of copies sold. Find the book
   - and update its quantity. Handle out-of-stock errors.
   - **Generate Low-Stock Report**: Display the details of all books where the quantity has fallen below 5.
3. **Functions**: Each menu option must be a separate function. To modify the inventory, you must pass the parallel arrays to the variable tracking the current number of books to these functions.

**Question No. 2**

A supermarket wants to manage its inventory and invoices generated each day to have a clear look at the stock and purchases. To increase code maintainability, multiple functions have been made for each functionality. Make efficient use of conditional statements, loops, arrays and functions, wherever necessary, to fulfil the requirements mentioned below.

Write a neat and organized C code for a menu driven application in which a user is free to choose and perform any of the following options:

- **Customer information:**
  - A function that asks the user to enter the name and CNIC number of the customer.
- **Display inventory**
  - This function will list all the products along with their quantity in stock as follows:

| Product code | Quantity in stock | Price per product |
|---|---|---|
| 001 | 50 | 100 |
| 002 | 10 | 200 |
| 003 | 20 | 300 |
| 004 | 8 | 150 |

- **Update inventory**
  - Whenever a product is purchased, its quantity in stock should be updated timely
- **Add item to cart**
  - A user can add items to the cart along with their quantity.
  - Update inventory
- **Display total bill**
  - Once the order is done, a user can proceed to check out and display the total bill.
  - Ask for promocodes. We are offering a promocode of Eid2025, ask the customer if he/she has a voucher then apply 25% discount on the total bill
- **Show invoice**
  - Display the customer details and the final bill with/without discount.
  - Display the bill with and without discount
- **Exit the system**

**Question No. 3**

You are a junior engineer at IESCO, tasked with programming a simulation for the power grid monitoring system in the Islamabad Capital Territory. To optimize memory, you must use a single integer for each sector in the grid to store multiple status flags by manipulating its individual bits.

**Core Requirements:**

1. **Grid Representation**: Use a 2D array, to represent the grid.
2. **Status Flags (Bitwise Logic)**: The status of each sector is encoded into the bits of its integer value:
   - Bit 0: Power Status (1 = ON, 0 = OFF)
   - Bit 1: Overload Warning (1 = Overloaded, 0 = Normal)
   - Bit 2: Maintenance Required (1 = Yes, 0 = No) A status of 3 (binary 0011) means Power is ON and there's an Overload Warning.
3. **Operator Interface**: Create a menu for an operator to:
   - Update Sector Status: Specify coordinates (row, col) and set or clear a specific status flag.
   - Query Sector Status: Query a sector's coordinates and print a human readable report of its status. o Run System Diagnostic: Scan the entire grid using nested loops and report the total number of sectors that are currently overloaded or require maintenance.
4. **Functions**: Encapsulate all primary operations (update, query, diagnostic) in functions that take the 2D array as a parameter.

**Question 4:**

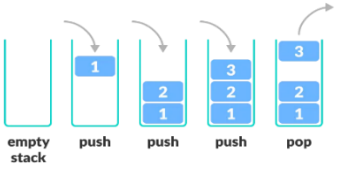
Implement a Stack using arrays and functions.

Your program should support the following operations through a menu-driven interface:

- **PUSH**: Insert an element into the stack
- **POP:** Remove the top element from the stack
- **PEEK**: Display the top element without removing it
- **DISPLAY**: Show all elements currently in the stack
- **EXIT**: Terminate the program

The stack should be implemented using a 1D array and each operation should have a separate user-defined functions (function protype is given just for the reference, you can opt any other function type if you want):

1. void push(int stack[], int top, int maxSize);
2. void pop(int stack[], int top);
3. void peek(int stack[], int top);
4. void display(int stack[], int top);




You should also check for the overflow and underflow conditions in case of PUSH and POP operations.

**Question No. 5**

TCS (Tranzum Courier Service), one of Pakistan's leading logistics companies, operates a network of dispatch centers and delivery riders spread across multiple cities. The dispatch center in Karachi coordinates daily deliveries and pickups by sending out route instructions, tracking updates, and sensitive customer information to its field riders.

However, these messages are often transmitted through unsecured digital channels — meaning that if someone intercepts the message, they could view addresses, package IDs, or delivery routes.

To enhance security without the need for expensive encryption systems, TCS decided to introduce a lightweight command-line tool that can quickly encode and decode text messages using a simple but consistent algorithm.

The goal of this project is to design and develop a simple C-based command-line utility that allows dispatch officers and riders to:

- **Encode** ordinary text messages into a coded format before transmission.
- **Decode** received coded messages back into readable text.

The encoding process must be reversible, ensuring no loss of information. The tool is intended for internal communication, not for high end cryptographic security, but for basic confidentiality and tamper resistance.

**Encoding Algorithm:**

When a dispatcher types a plain text message (e.g., "Rider ready for delivery"), the program applies the following two-step transformation:

1. Reversal Step
   - The entire message string is reversed character by character.
   - Example: "Rider ready for delivery" → "yreviled rof ydaer rediR"
2. Bit Toggling Step
   - Each character in the reversed string is converted to its binary representation (ASCII value).
   - Then, the program toggles the 2nd and 5th bits of this value.
     Toggling means: if the bit is 1, make it 0; if it's 0, make it 1.
   - The modified binary value is converted back to a new encoded character.
   - This creates a transformed message that looks like random symbols or gibberish to anyone intercepting it.

After these steps, the encoded message is displayed and ready to be sent securely over text or radio.

**Decoding Algorithm**

On the receiver's side (the delivery rider's device), the decoding process must perfectly reverse the encoding.

The decoding algorithm applies the same steps in reverse order:

1. Bit Reversal (Untoggling):
   - The program takes the encoded text and toggles the 2nd and 5th bits of each character again.
   - Since toggling twice restores the original bits, this effectively undoes the first transformation.
2. String Reversal:
   - The resulting string is then reversed again to restore the original order of characters.

After these two steps, the original message text (e.g., "Rider ready for delivery") is recovered and displayed on screen.

**Program Flow**

The program follows an **interactive command-line flow** for ease of use by dispatch staff:

1. Display a simple **menu**:
   - Encode Message
   - Decode Message
   - Exit
2. Based on the user's choice:
   - The program asks for the message input.
   - It then calls the appropriate function:
     - void encodeMessage(char message[])
     - void decodeMessage(char message[])
   - Each function performs its respective operation using string and bitwise logic.
3. The result is displayed back on screen
4. The user can continue encoding/decoding multiple messages until they choose to exit.