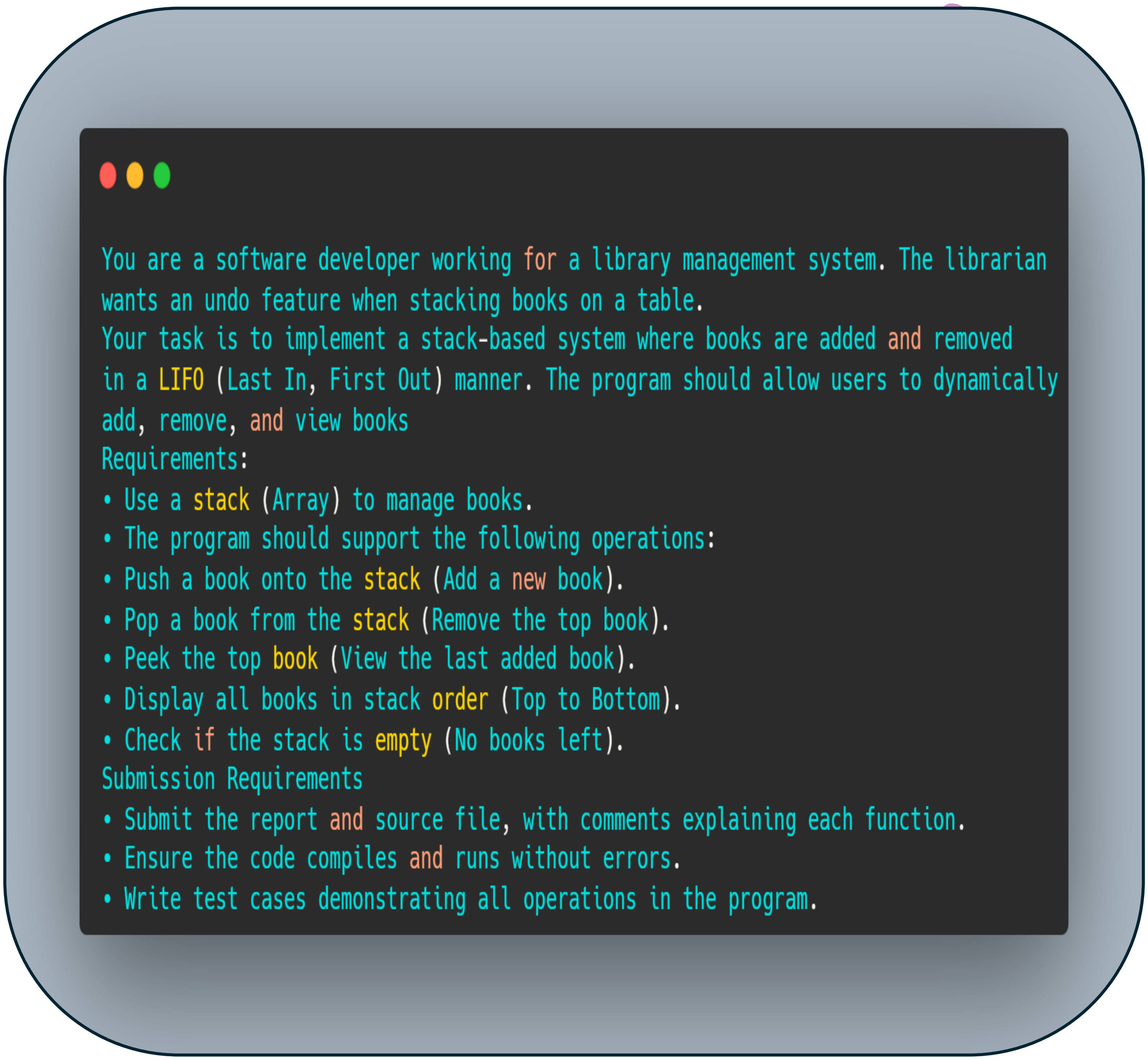
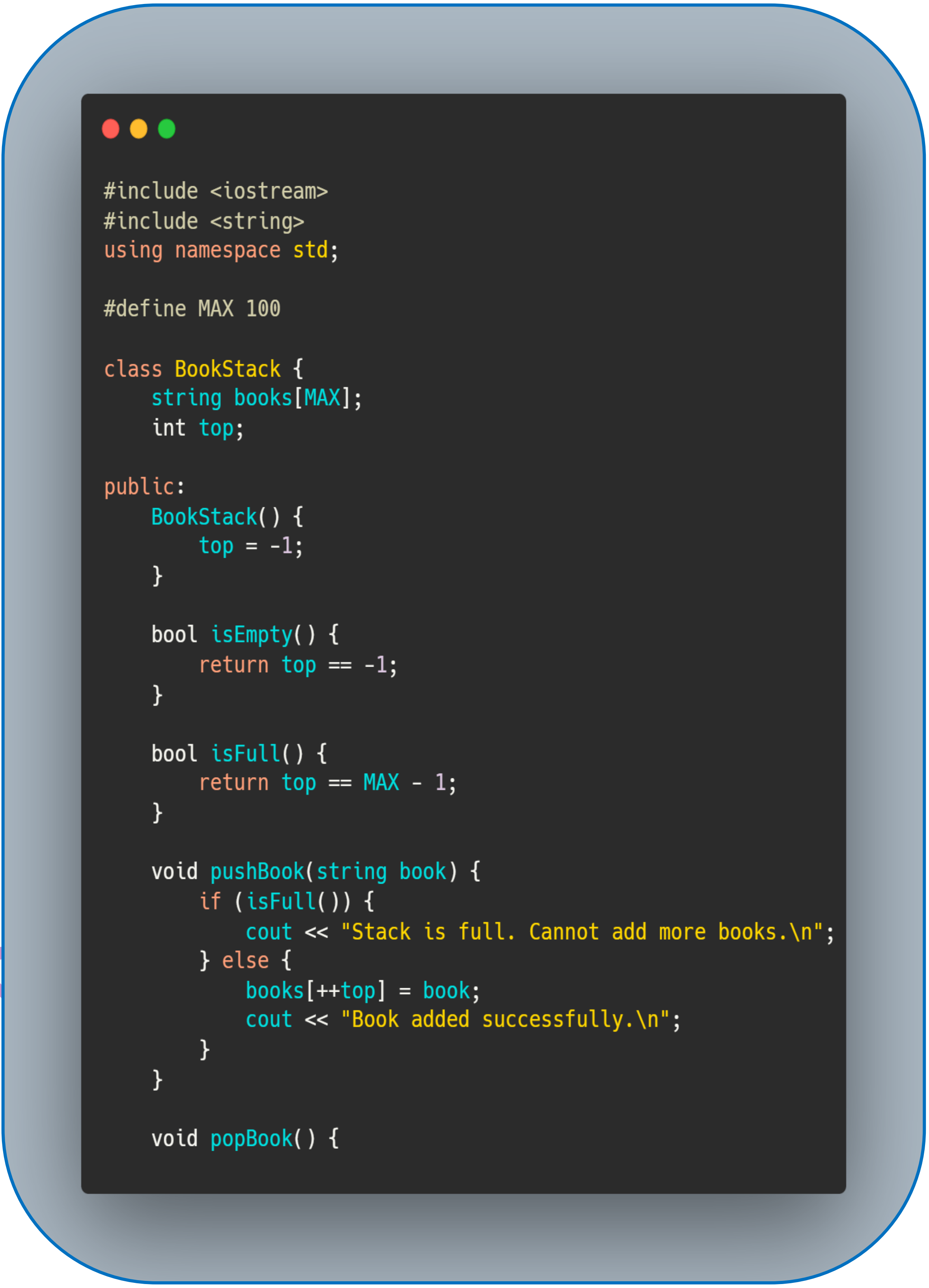
Lab Task 8 [Book Management System]

Task:

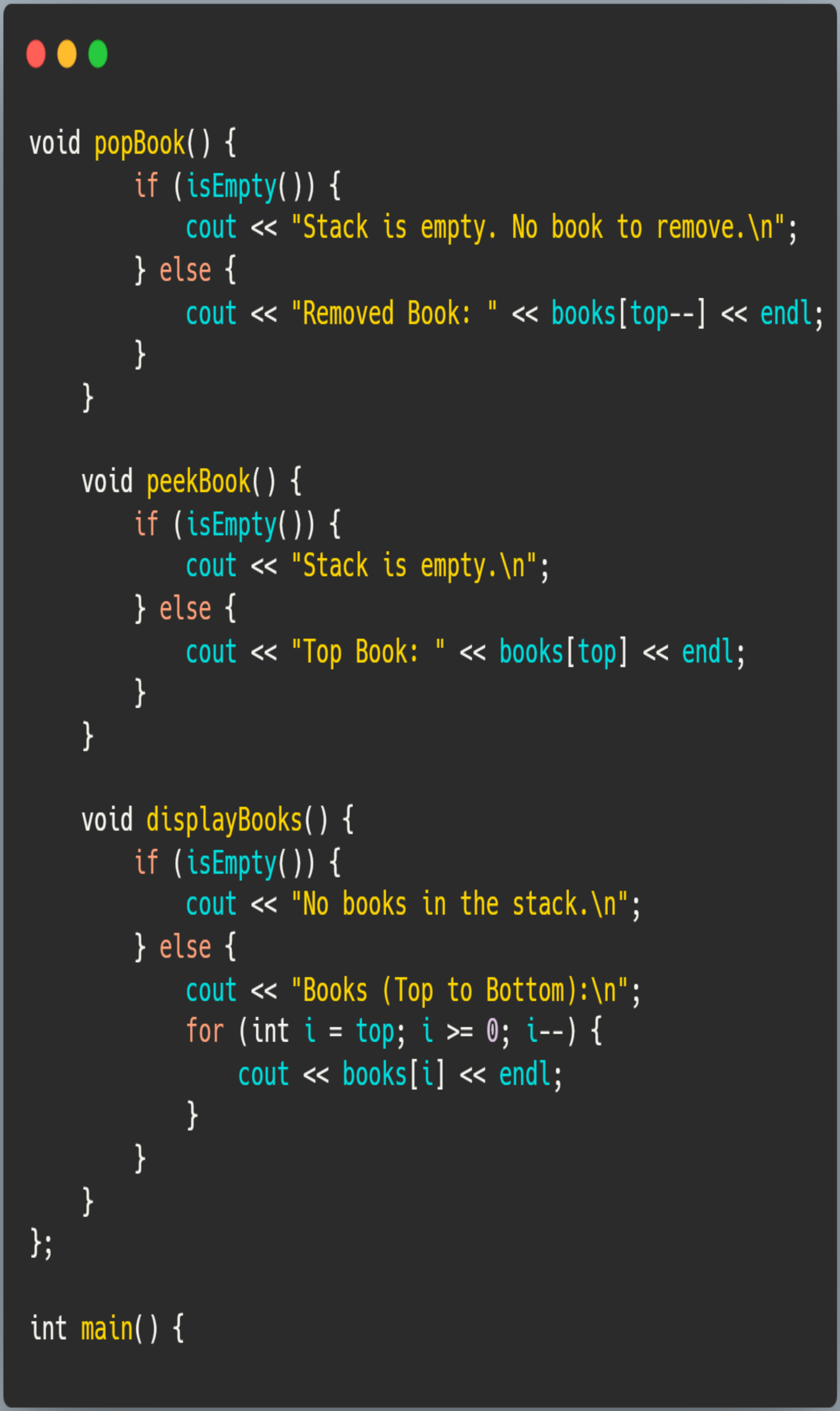


```
You are a software developer working for a library management system. The librarian
wants an undo feature when stacking books on a table.
Your task is to implement a stack-based system where books are added and removed
in a LIFO (Last In, First Out) manner. The program should allow users to dynamically
add, remove, and view books
Requirements:
• Use a stack (Array) to manage books.
• The program should support the following operations:
• Push a book onto the stack (Add a new book).
• Pop a book from the stack (Remove the top book).
• Peek the top book (View the last added book).
• Display all books in stack order (Top to Bottom).
• Check if the stack is empty (No books left).
Submission Requirements
• Submit the report and source file, with comments explaining each function.
• Ensure the code compiles and runs without errors.
• Write test cases demonstrating all operations in the program.
```

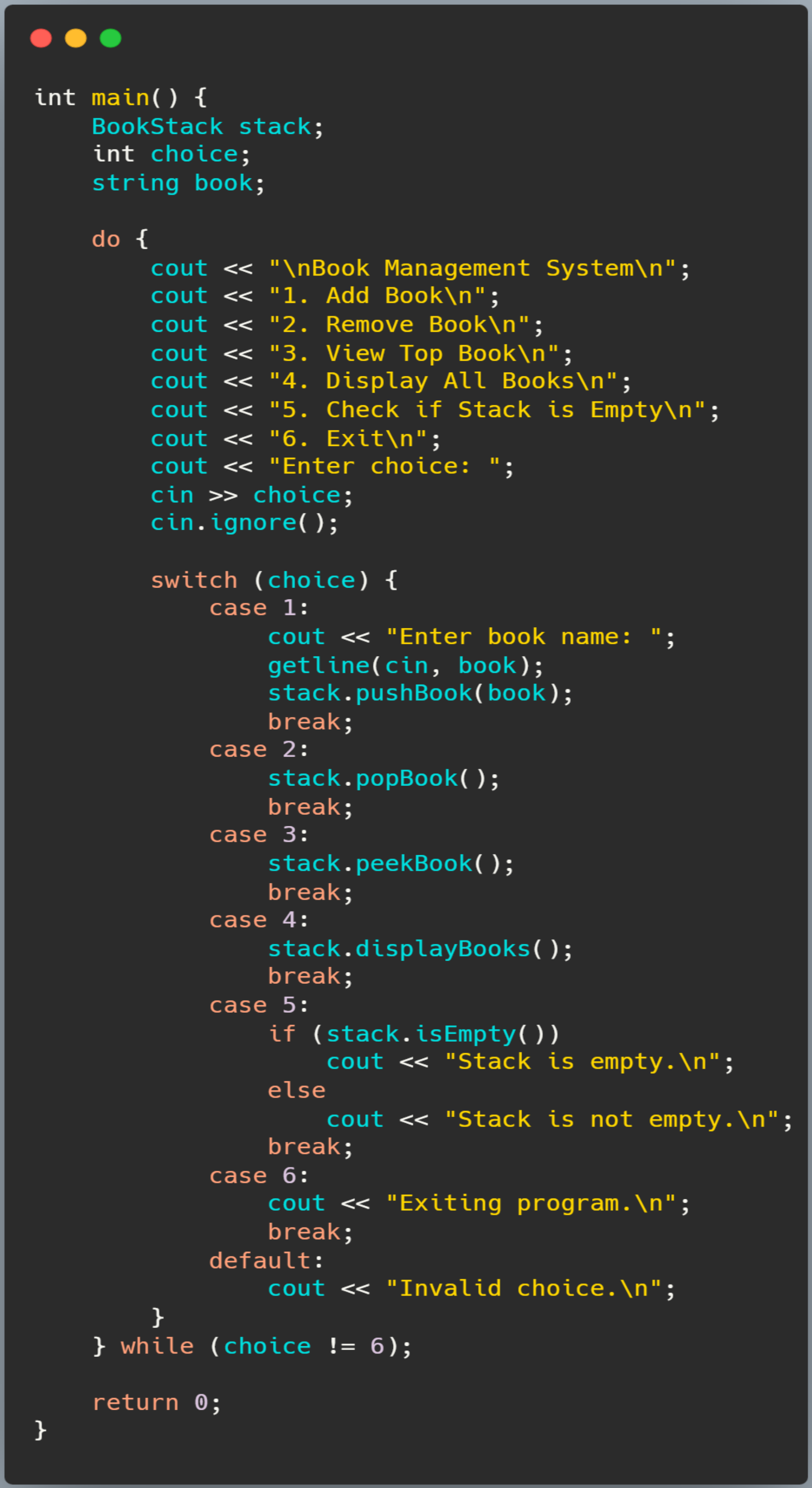
Code:



```
#include <iostream>
#include <string>
using namespace std;

#define MAX 100

class BookStack {
    string books[MAX];
    int top;

public:
    BookStack() {
        top = -1;
    }

    bool isEmpty() {
        return top == -1;
    }

    bool isFull() {
        return top == MAX - 1;
    }

    void pushBook(string book) {
        if (isFull()) {
            cout << "Stack is full. Cannot add more books.\n";
        } else {
            books[++top] = book;
            cout << "Book added successfully.\n";
        }
    }

    void popBook() {
```



```cpp
void popBook() {
        if (isEmpty()) {
            cout << "Stack is empty. No book to remove.\n";
        } else {
            cout << "Removed Book: " << books[top--] << endl;
        }
    }

    void peekBook() {
        if (isEmpty()) {
            cout << "Stack is empty.\n";
        } else {
            cout << "Top Book: " << books[top] << endl;
        }
    }

    void displayBooks() {
        if (isEmpty()) {
            cout << "No books in the stack.\n";
        } else {
            cout << "Books (Top to Bottom):\n";
            for (int i = top; i >= 0; i--) {
                cout << books[i] << endl;
            }
        }
    }
};

int main() {
```

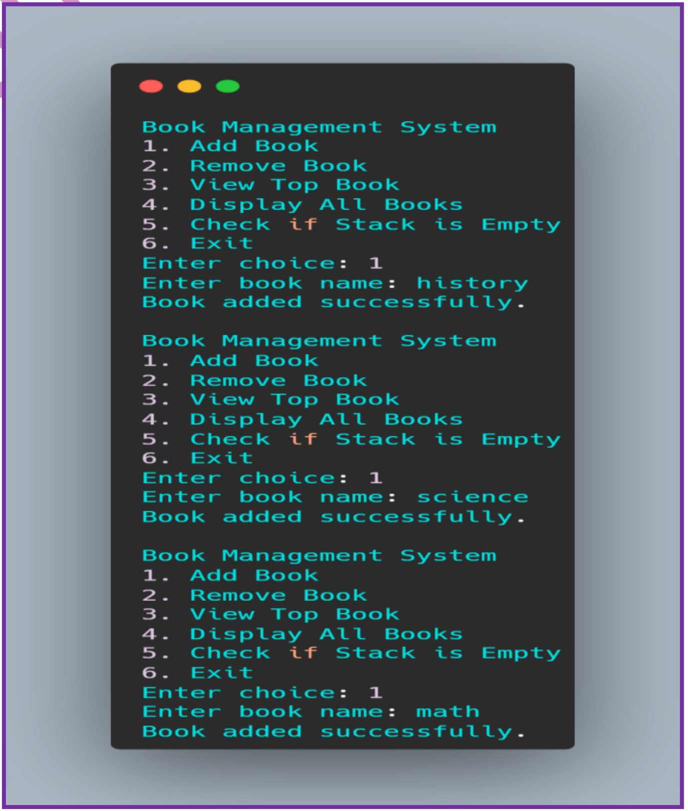


```cpp
int main() {
    BookStack stack;
    int choice;
    string book;

    do {
        cout << "\nBook Management System\n";
        cout << "1. Add Book\n";
        cout << "2. Remove Book\n";
        cout << "3. View Top Book\n";
        cout << "4. Display All Books\n";
        cout << "5. Check if Stack is Empty\n";
        cout << "6. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1:
                cout << "Enter book name: ";
                getline(cin, book);
                stack.pushBook(book);
                break;
            case 2:
                stack.popBook();
                break;
            case 3:
                stack.peekBook();
                break;
            case 4:
                stack.displayBooks();
                break;
            case 5:
                if (stack.isEmpty())
                    cout << "Stack is empty.\n";
                else
                    cout << "Stack is not empty.\n";
                break;
            case 6:
                cout << "Exiting program.\n";
                break;
            default:
                cout << "Invalid choice.\n";
        }
    } while (choice != 6);

    return 0;
}
```

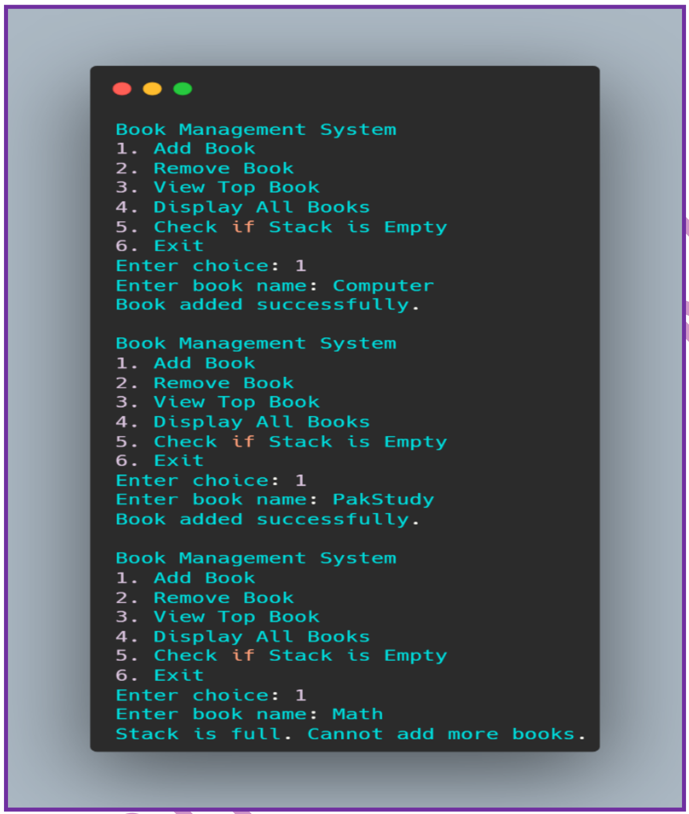
Table Of Output:

	Condition	Output
1	Push book when stack is **not full**	Book added successfully.
2	Push book when stack is **full**	Stack is full. Cannot add more books.
3	Pop book when stack is **not empty**	Removed Book: <book name>
4	Pop book when stack is **empty**	Stack is empty. No book to remove.
5	Peek book when stack is **not empty**	Top Book: <book name>
6	Peek book when stack is **empty**	Stack is empty.
7	Display books when stack is **not empty**	Books (Top to Bottom) printed
8	Display books when stack is **empty**	No books in the stack.
9	Check empty when stack **is empty**	Stack is empty.
10	Check empty when stack **is not empty**	Stack is not empty.
11	User enters **Exit (6)**	Exiting program.
12	User enters **invalid menu choice**	Invalid choice.
	Total 12 Possible Outputs	

Push book when stack is **not full**

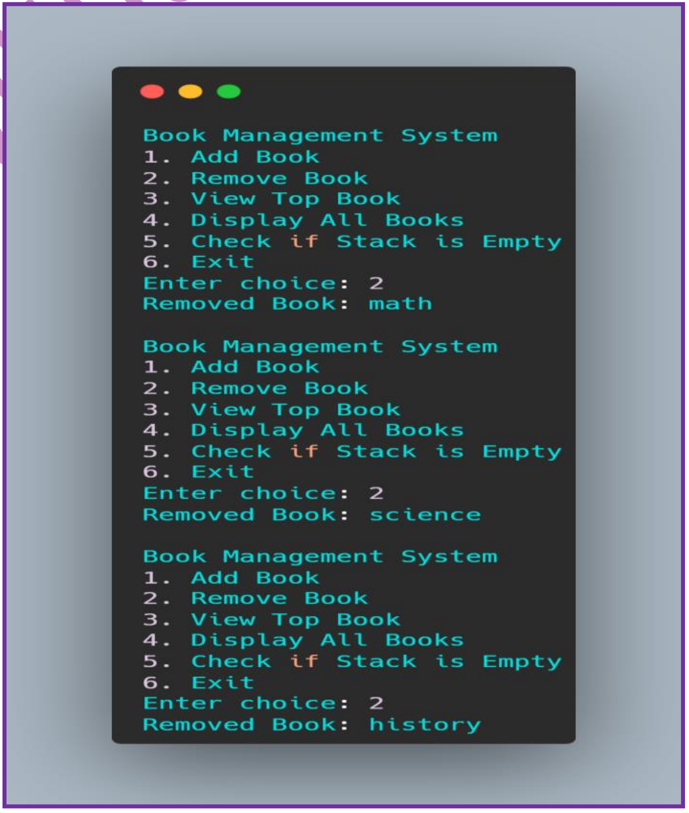

```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: history
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: science
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: math
Book added successfully.
```

Push book when stack is **full**



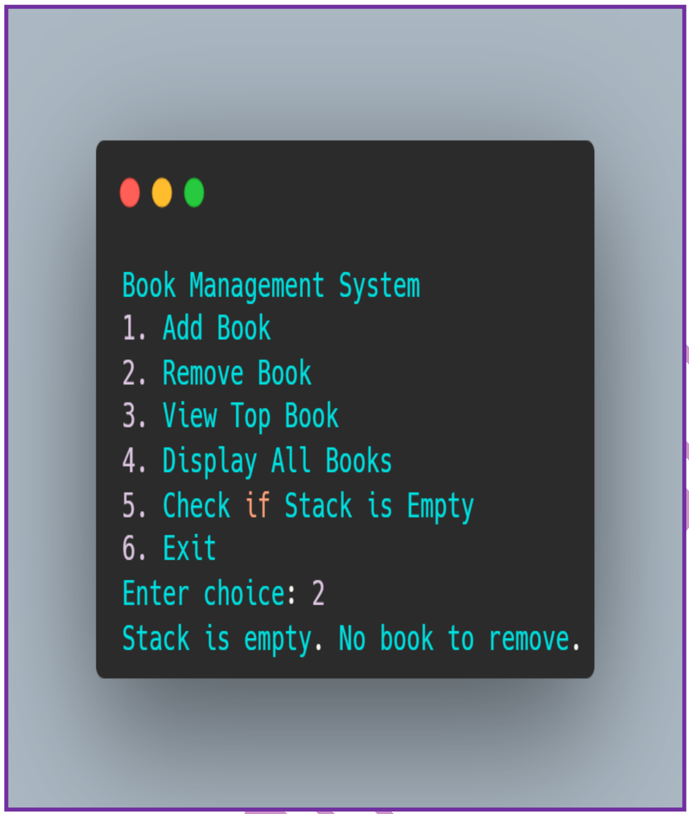
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: Computer
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: PakStudy
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: Math
Stack is full. Cannot add more books.
```

Pop book when stack is **not empty**



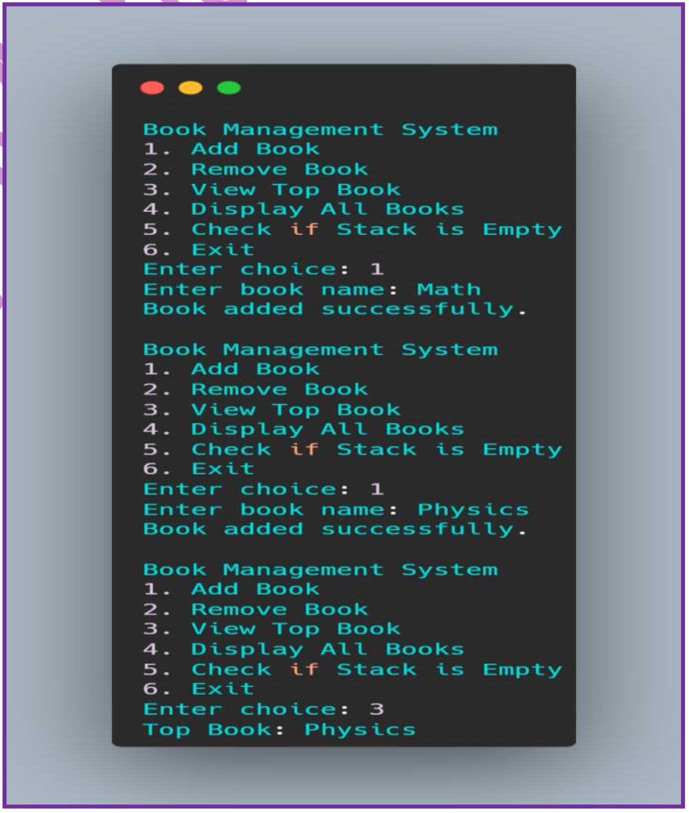
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 2
Removed Book: math

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 2
Removed Book: science

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 2
Removed Book: history
```

Pop book **when** stack is **empty**



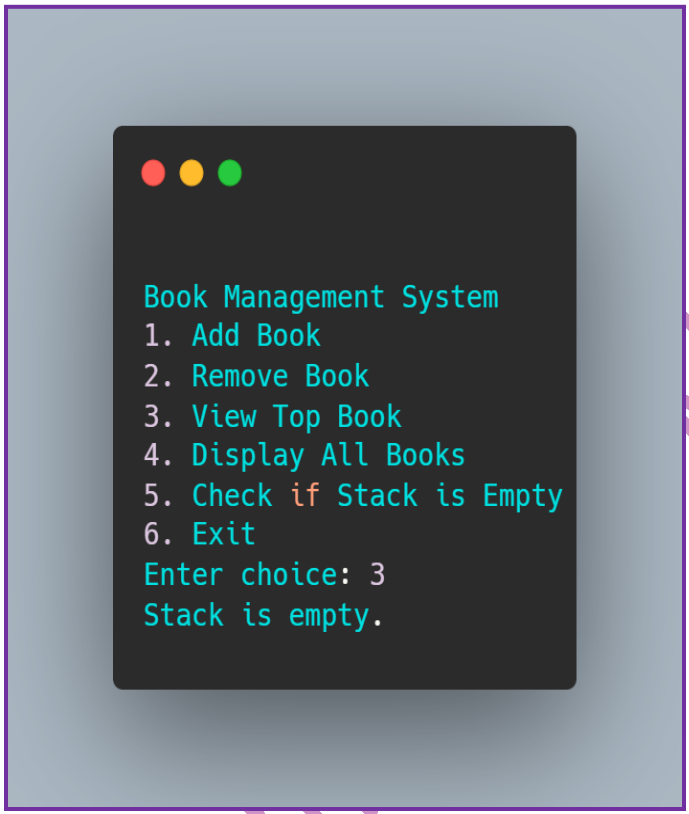
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 2
Stack is empty. No book to remove.
```

Peek book **when** stack is **not empty**



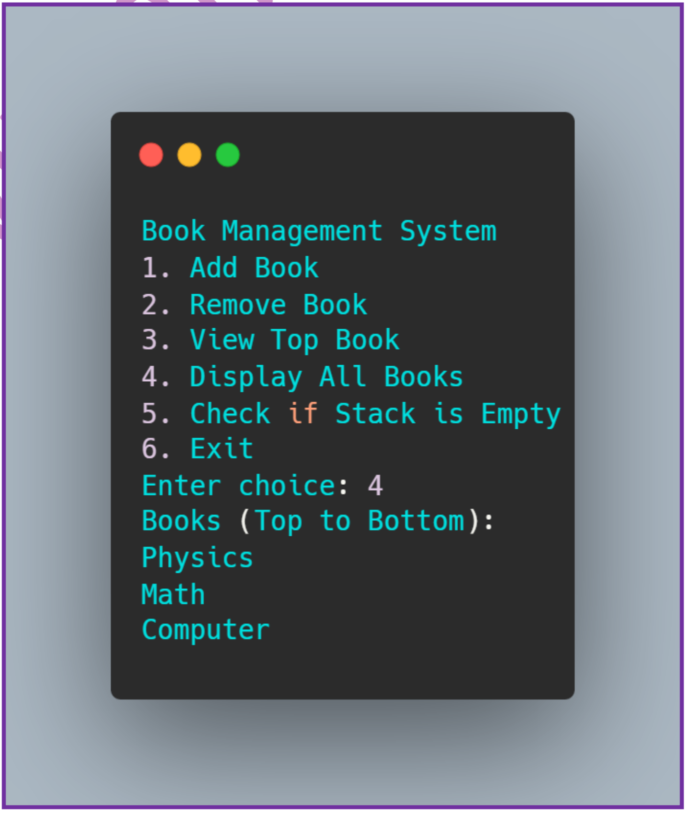
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: Math
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 1
Enter book name: Physics
Book added successfully.

Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 3
Top Book: Physics
```

Peek book when stack is **empty**



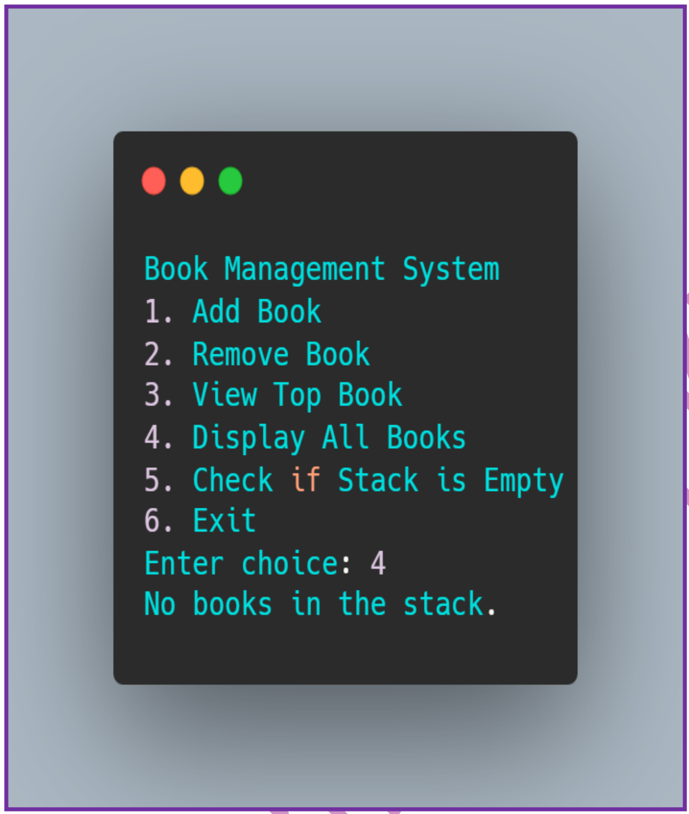
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 3
Stack is empty.
```

Display books when stack is **not empty**



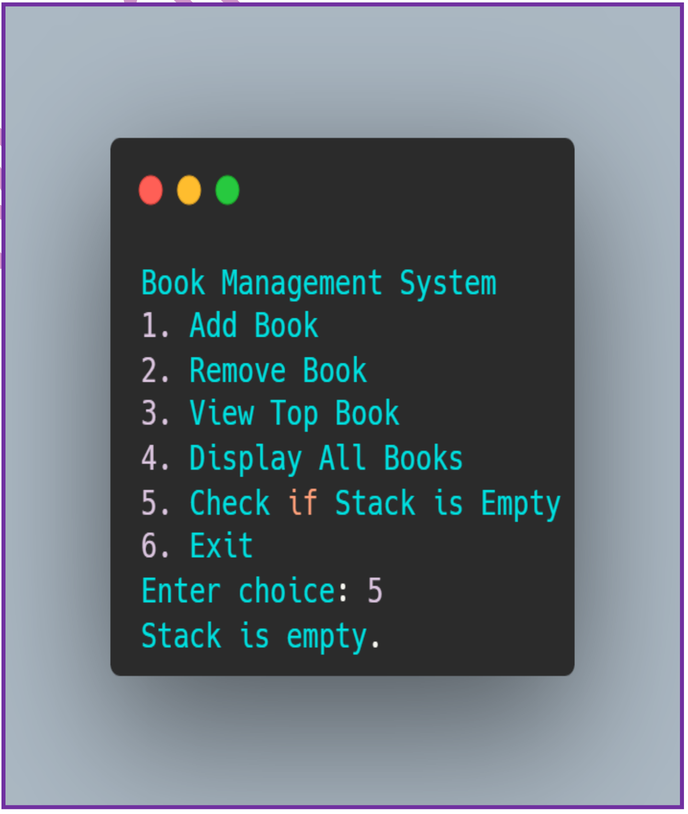
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 4
Books (Top to Bottom):
Physics
Math
Computer
```

Display books **when** stack is **empty**



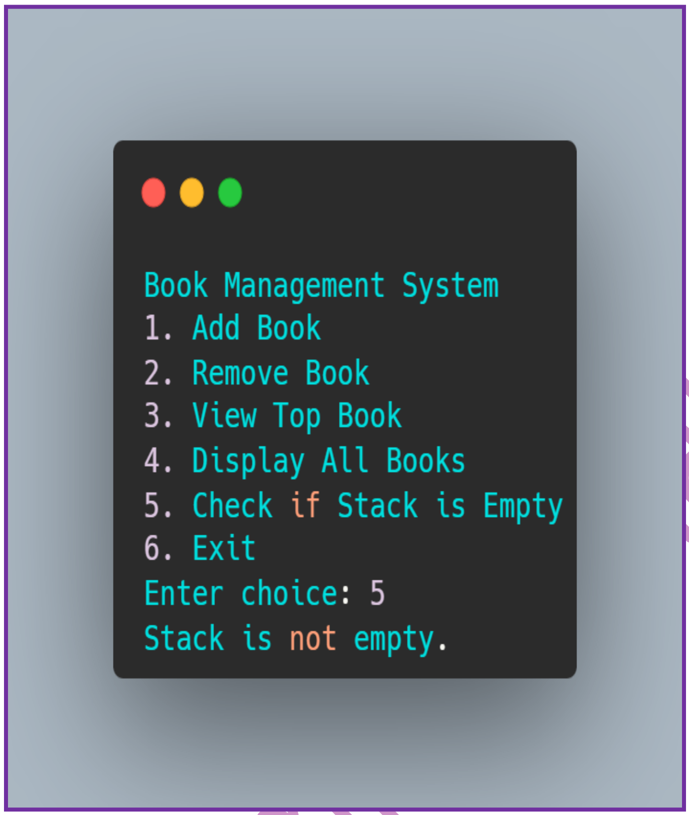
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 4
No books in the stack.
```

Check empty **when** stack **is empty**



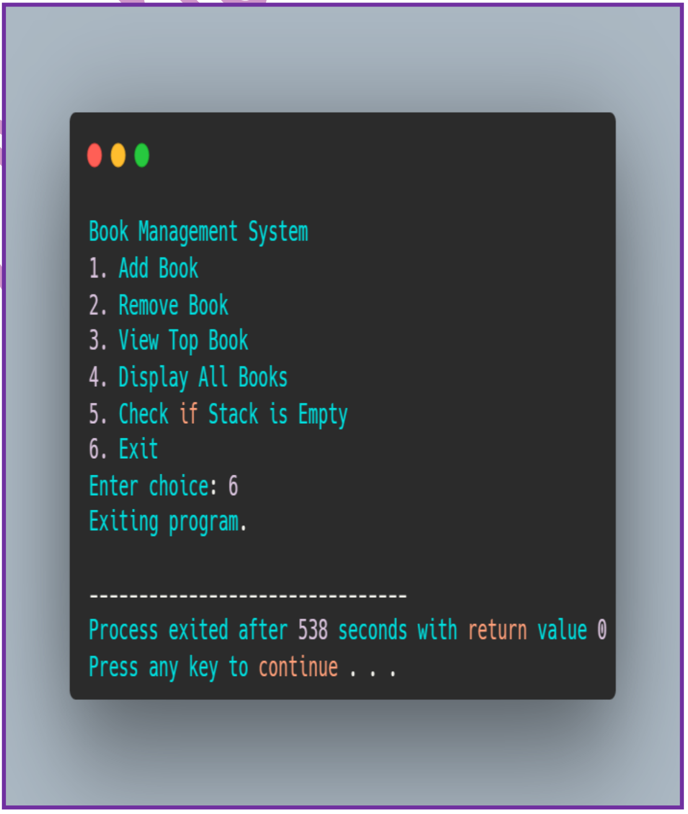
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 5
Stack is empty.
```

Check empty **when** stack **is not empty**



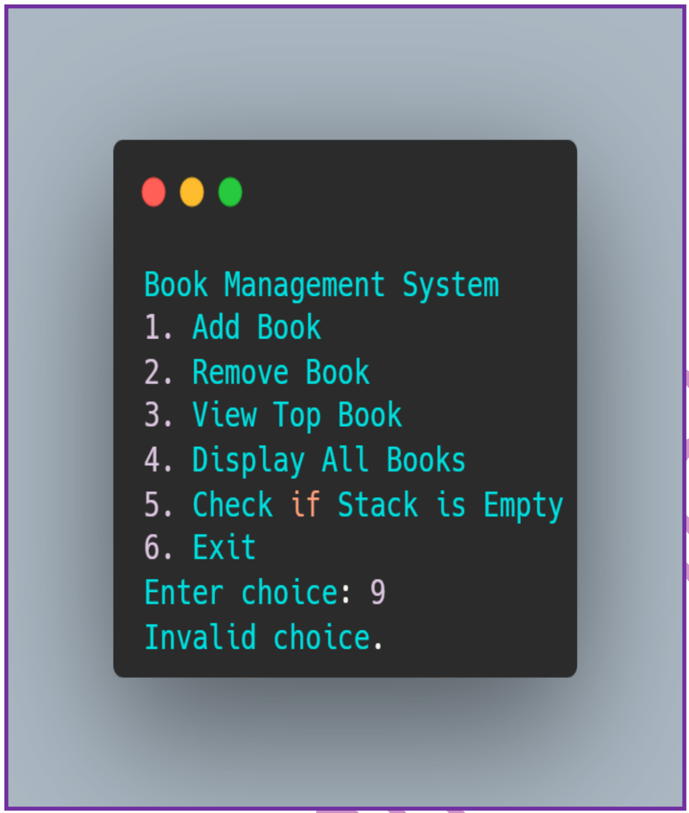
```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 5
Stack is not empty.
```

When User enters **Exit (6)**



```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 6
Exiting program.


--------------------------------
Process exited after 538 seconds with return value 0
Press any key to continue . . .
```

When invalid menu choice



```
Book Management System
1. Add Book
2. Remove Book
3. View Top Book
4. Display All Books
5. Check if Stack is Empty
6. Exit
Enter choice: 9
Invalid choice.
```

Click here to Get this code on GitHub

Click here to Test this Code by Yourself.

Conclusion: -

The Book Management System implemented using a stack can generate **12 distinct logical outputs**, determined by user choices and the current state of the stack (empty, partially filled, or full).