

**Assignment #1(Spring-2024)**

**(Object Oriented Programming)**

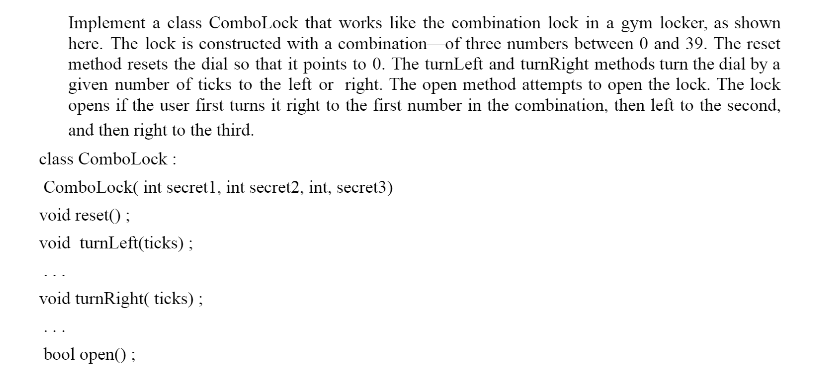
**First year(2nd Semester)**

**Software Engineering Department**

**NED University of Engineering and Technology**

**CLO:2 Submission Date:10th June,24**

**Question #1**



**CODE:**

#include <iostream>

using namespace std;

class ComboLock

{

private:

    int secret1, secret2, secret3;

    int currentPosition;

    int step;

    int input1, input2, input3;

public:

    ComboLock(int secret1, int secret2, int secret3)

    {

        this->secret1 = secret1;

        this->secret2 = secret2;

        this->secret3 = secret3;

        reset();

    }

    void reset()

    {

        currentPosition = 0;

        step = 0;

        input1 = input2 = input3 = -1;

    }

    void turnLeft(int ticks)

    {

        currentPosition = (currentPosition - ticks + 40) % 40;

        if (step == 1)

        {

            input2 = currentPosition;

            step++;

        }

        cout << "Turned Left to " << currentPosition << endl;

    }

    void turnRight(int ticks)

    {

        currentPosition = (currentPosition + ticks) % 40;

        if (step == 0)

        {

            input1 = currentPosition;

            step++;

        }

        else if (step == 2)

        {

            input3 = currentPosition;

            step++;

        }

        cout << "Turned Right to " << currentPosition << endl;

    }

    bool open()

    {

        if (input1 == secret1 && input2 == secret2 && input3 == secret3)

        {

            cout << "Lock opened successfully!" << endl;

            return true;

        }

        else

        {

            cout << "Failed to open the lock." << endl;

            return false;

        }

    }

};

int main()

{

    ComboLock lock(10, 20, 30);

    lock.reset();

    lock.turnRight(10);

    lock.turnLeft(30);

    lock.turnRight(10);

    lock.open();

    lock.reset();

    lock.turnRight(5);

    lock.turnLeft(15);

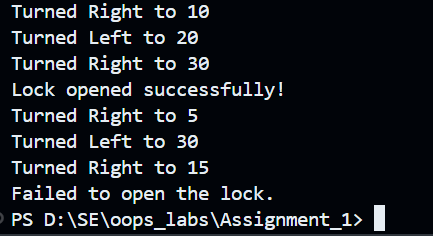
    lock.turnRight(25);

    lock.open();

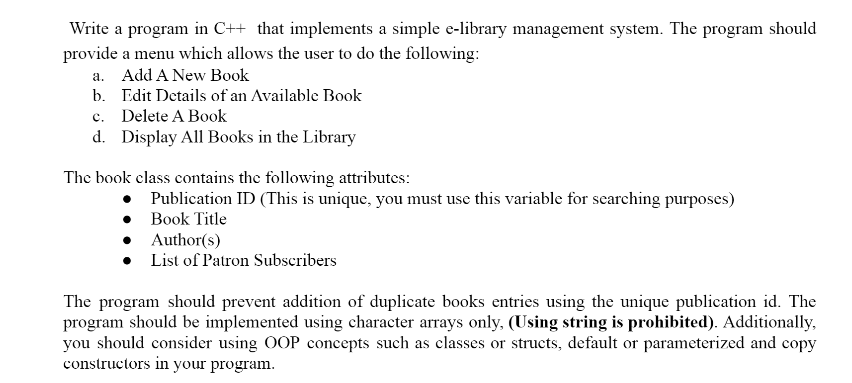
    return 0;

}

**OUTPUT:**



**Question #2**



**CODE:**

#include <iostream>

#include <cstring>

using namespace std;

class Book

{

public:

    char publicationID[20];

    char title[50];

    char author[50];

    char patrons[200];

    Book()

    {

        strcpy(publicationID, "");

        strcpy(title, "");

        strcpy(author, "");

        strcpy(patrons, "");

    }

    Book(const char \*pubID, const char \*bookTitle, const char \*bookAuthor, const char \*bookPatrons)

    {

        strcpy(publicationID, pubID);

        strcpy(title, bookTitle);

        strcpy(author, bookAuthor);

        strcpy(patrons, bookPatrons);

    }

    void editBookDetails(const char \*newTitle, const char \*newAuthor, const char \*newPatrons)

    {

        strcpy(title, newTitle);

        strcpy(author, newAuthor);

        strcpy(patrons, newPatrons);

    }

};

class Library

{

private:

    Book books[100];

    int bookCount;

public:

    Library() : bookCount(0) {}

    bool addBook(const Book &newBook)

    {

        for (int i = 0; i < bookCount; ++i)

        {

            if (strcmp(books[i].publicationID, newBook.publicationID) == 0)

            {

                cout << "Book with this Publication ID already exists!" << endl;

                return false;

            }

        }

        books[bookCount++] = newBook;

        return true;

    }

    bool editBook(const char \*pubID, const char \*newTitle, const char \*newAuthor, const char \*newPatrons)

    {

        for (int i = 0; i < bookCount; ++i)

        {

            if (strcmp(books[i].publicationID, pubID) == 0)

            {

                books[i].editBookDetails(newTitle, newAuthor, newPatrons);

                return true;

            }

        }

        cout << "Book not found!" << endl;

        return false;

    }

    bool deleteBook(const char \*pubID)

    {

        for (int i = 0; i < bookCount; ++i)

        {

            if (strcmp(books[i].publicationID, pubID) == 0)

            {

                for (int j = i; j < bookCount - 1; ++j)

                {

                    books[j] = books[j + 1];

                }

                bookCount--;

                return true;

            }

        }

        cout << "Book not found!" << endl;

        return false;

    }

    void displayBooks() const

    {

        for (int i = 0; i < bookCount; ++i)

        {

            cout << "Publication ID: " << books[i].publicationID << endl;

            cout << "Title: " << books[i].title << endl;

            cout << "Author: " << books[i].author << endl;

            cout << "Patrons: " << books[i].patrons << endl;

            cout << "-------------------------------------" << endl;

        }

    }

};

int main()

{

    Library library;

    int choice;

    char pubID[20], title[50], author[50], patrons[200];

    while (true)

    {

        cout << "\nE-Library Management System\n";

        cout << "1. Add A New Book\n";

        cout << "2. Edit Details of an Available Book\n";

        cout << "3. Delete A Book\n";

        cout << "4. Display All Books in the Library\n";

        cout << "5. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        cin.ignore();

        switch (choice)

        {

        case 1:

            cout << "Enter Publication ID: ";

            cin.getline(pubID, 20);

            cout << "Enter Book Title: ";

            cin.getline(title, 50);

            cout << "Enter Author: ";

            cin.getline(author, 50);

            cout << "Enter Patrons: ";

            cin.getline(patrons, 200);

            if (library.addBook(Book(pubID, title, author, patrons)))

            {

                cout << "Book added successfully!" << endl;

            }

            break;

        case 2:

            cout << "Enter Publication ID of the book to edit: ";

            cin.getline(pubID, 20);

            cout << "Enter New Title: ";

            cin.getline(title, 50);

            cout << "Enter New Author: ";

            cin.getline(author, 50);

            cout << "Enter New Patrons: ";

            cin.getline(patrons, 200);

            if (library.editBook(pubID, title, author, patrons))

            {

                cout << "Book details updated successfully!" << endl;

            }

            break;

        case 3:

            cout << "Enter Publication ID of the book to delete: ";

            cin.getline(pubID, 20);

            if (library.deleteBook(pubID))

            {

                cout << "Book deleted successfully!" << endl;

            }

            break;

        case 4:

            library.displayBooks();

            break;

        case 5:

            return 0;

        default:

            cout << "Invalid choice! Please try again." << endl;

        }

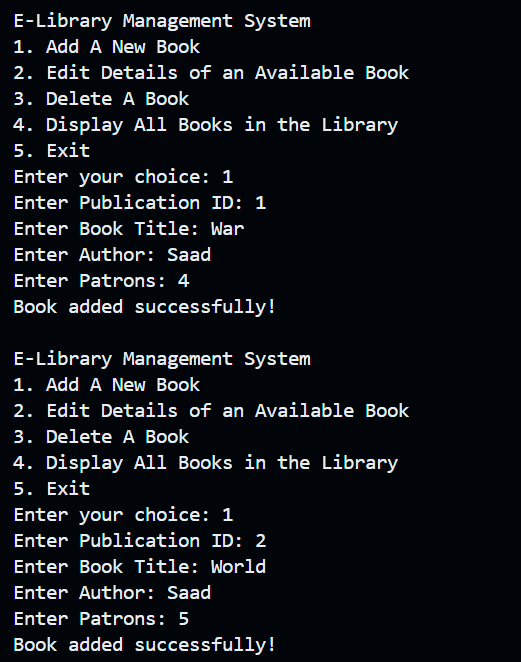
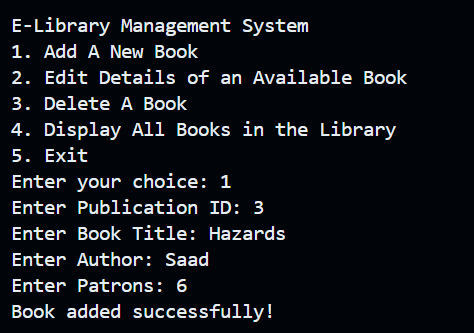
    }

    return 0;

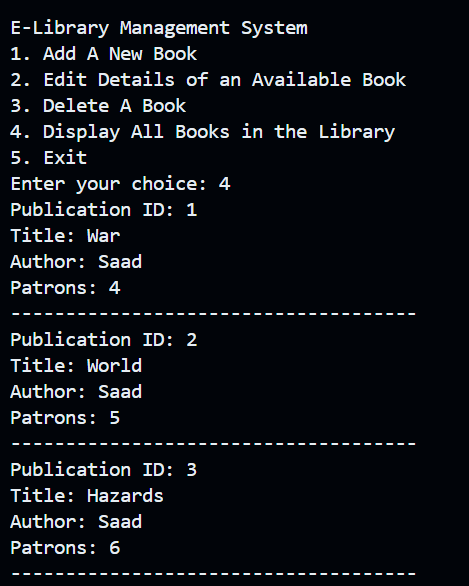
}

**OUTPUT:**

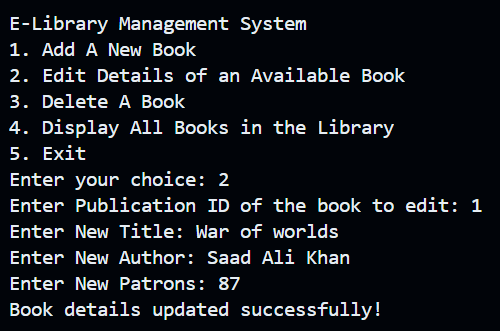
**ADD BOOK:**

****

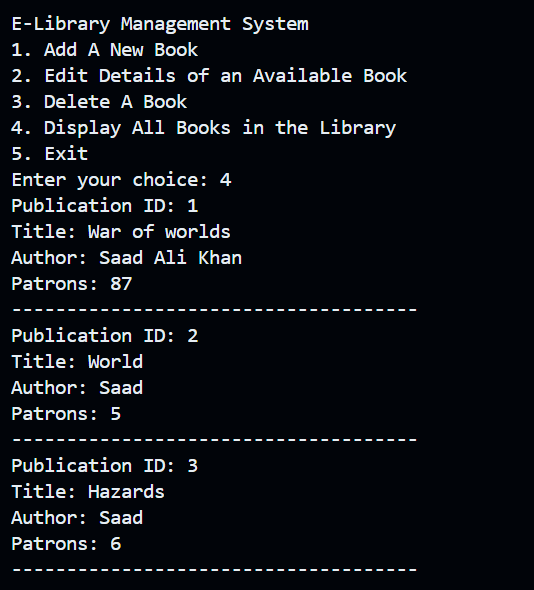
**DISPLAY ALL BOOKS:**

****

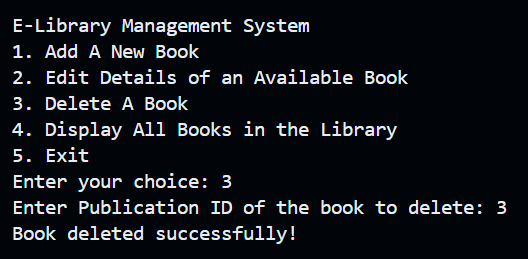
**EDIT DETAILS OF A BOOK:**

****

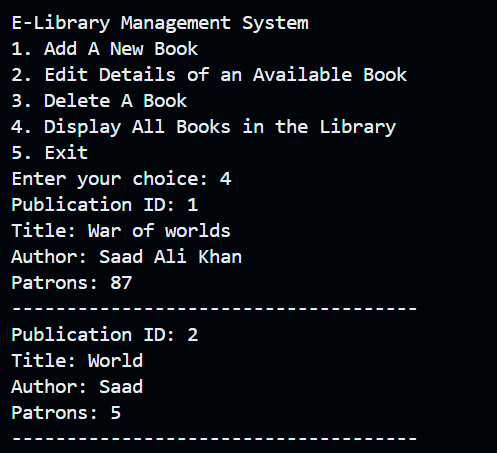
NOTE: Now when we again display all books, book with id 1 will have updated data.



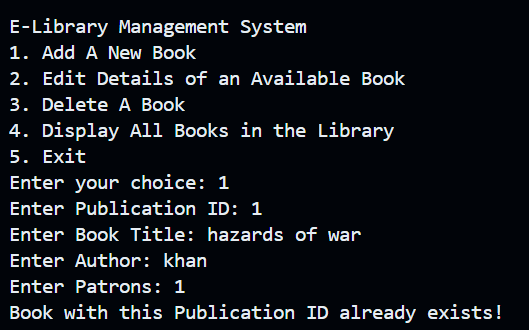
**DELETE A BOOK:**

****

NOTE: Now when we again display all books, book with id 3 will be deleted.



**NO DUPLICATE OF A BOOK:**

****

NOTE: Since the library already has a book with id 1 so another book with same id cannot be added

