Principles of Programming Language (CS302)

Assignment - 2

**U19CS012**

1.) Create a Program with function update having parameters as int \*a & int \*b.

Modify the values in memory so that **a** - > their sum and **b** -> their absolute difference.

Basic Concepts of **Pointers**

|  |  |
| --- | --- |
| Symbol | Purpose |
| &val | the memory address of val |
| int \*p = &val | assigns the memory address of val to pointer p |
| (\*p) | will return the value stored in val and any modification to it will be performed on val. |

**Problem**

Given a, b return **a’, b’** such that:

**a' = a + b**

**b' = |a - b|**

**Code**

*#include* <iostream>

*#include* <iostream>

using namespace std;

*// U19CS012 [BHAGYA VINOD RANA]*

*// Modifies the Values in a,b to a+b, absolute(a-b)*

void modify(int \*a, int \*b);

int main()

{

    int a, b;

*// addr\_a - stores the memory address of variable 'a'*

    int \*addr\_a = &a;

*// addr\_b - stores the memory address of variable 'b'*

    int \*addr\_b = &b;

    cout << "Enter Values of 'a' and 'b' : ";

    cin >> a >> b;

*// Function call to Modify the Values*

    modify(addr\_a, addr\_b);

    cout << "Modified Values [ a+b, |a-b|] : ";

    cout << a << " " << b << endl;

*return* 0;

}

*// Modifies the Values in a,b to a+b, absolute(a-b)*

void modify(int \*a, int \*b)

{

*// Method 1*

*// Store the Value of a in tmp variable*

    int tmp = \*a;

*// Modify a -> a + b*

    \*a = \*a + \*b;

*// b -> abs(a-b) [Note: We have stores the Intial Value of a in tmp]*

    \*b = tmp - \*b;

*// If val of b is negative, Multiply it by -1 to make it Positive*

*if* (\*b < 0)

        \*b = (\*b) \* (-1);

*// Short Method 2*

*// \*a = (\*a) + (\*b);*

*// Effective (a+b)-(2\*b) => a-b*

*// \*b = (\*a) - (2 \* (\*b));*

*// If val of b is negative, Multiply it by -1 to make it Positive*

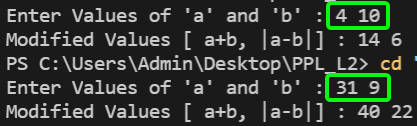
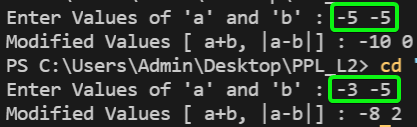
*// if (\*b < 0)*

*//     \*b = (\*b) \* (-1);*

*return*;

}

**Output**

2.) Write a program with two classes HotelRoom and HotelApartment denoting respectively a standard hotel room and a hotel apartment. An instance of any of these classes has two parameters: **bedrooms** and **bathrooms** denoting respectively *the number of bedrooms* and *the number of bathrooms in the room*.

The prices of a standard hotel room and hotel apartment are given as:

* **Hotel Room**: 50 x bedrooms + 100 x bathrooms.
* **Hotel Apartment**: (Standard Hotel Room Price) + 100.

*For example, if a standard room costs 200, then an apartment with the same number of bedrooms and bathrooms costs 300.*

Write a program to return the correct profit. Make necessary assumptions wherever necessary.

[Best Question to Understand the **Need for Virtual Function**]

**Code**

*// Question Link - https://www.hackerrank.com/challenges/hotel-prices/problem*

*#include* <iostream>

*#include* <vector>

*// U19CS012 [BHAGYA VINOD RANA]*

using namespace std;

*// HotelRoom Class*

class HotelRoom

{

private:

    int no\_of\_bedrooms;

    int no\_of\_bathrooms;

public:

*// HotelRoom Constructor*

    HotelRoom(int bedrooms, int bathrooms)

    {

        no\_of\_bedrooms = bedrooms;

        no\_of\_bathrooms = bathrooms;

    }

*//  virtual int get\_profit() ---(Instead Of)---> int get\_profit()*

    virtual int get\_profit()

    {

*return* ((50 \* no\_of\_bedrooms) + (100 \* no\_of\_bathrooms));

    }

};

*// HotelApartment Class*

class HotelApartment : public HotelRoom

{

public:

*// HotelApartment Constructor*

    HotelApartment(int bedrooms, int bathrooms) : HotelRoom(bedrooms, bathrooms) {}

*// We want the Derived Class get\_profit() to be Executed/ Overridden on Base Class get\_profit()*

    int get\_profit()

    {

*return* ((HotelRoom::get\_profit()) + 100);

    }

};

int main()

{

    int n;

    cin >> n;

    vector<HotelRoom \*> rooms;

    string room\_type;

    int bedrooms, bathrooms;

*for* (int i = 0; i < n; ++i)

    {

        cin >> room\_type >> bedrooms >> bathrooms;

*if* (room\_type == "standard")

            rooms.push\_back(new HotelRoom(bedrooms, bathrooms));

*else*

            rooms.push\_back(new HotelApartment(bedrooms, bathrooms));

    }

    int total\_profit = 0;

*// The Real Problem in this Code Sample [get\_profit()]*

*for* (auto room : rooms)

        total\_profit += room->get\_profit();

    cout << "Total Profits : " << total\_profit << endl;

*// Free Up Memory by Deallocating the Objects*

*for* (auto room : rooms)

    {

        delete room;

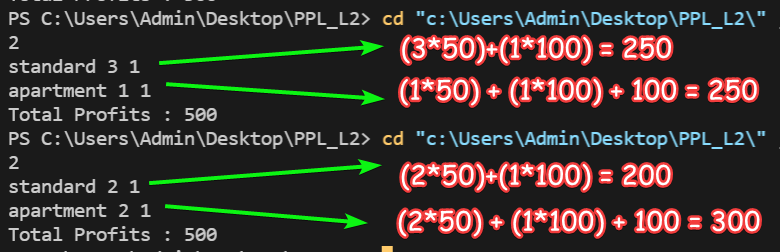
    }

    rooms.clear();

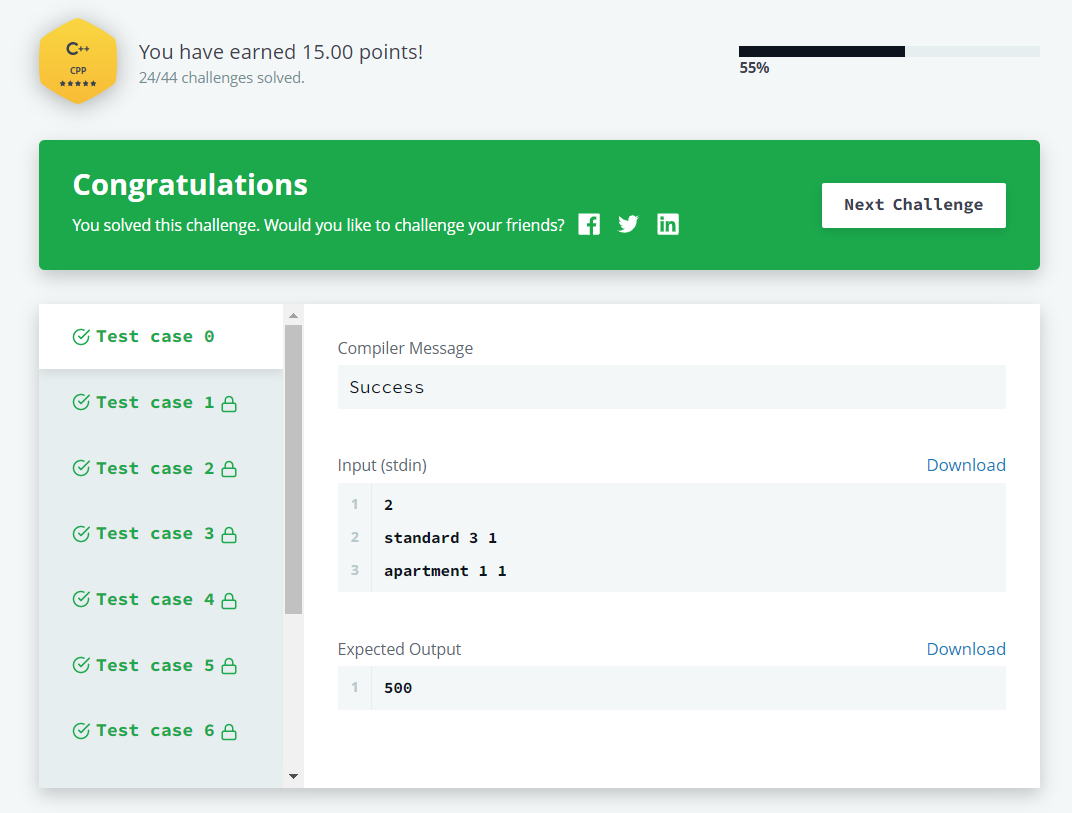
*return* 0;

**Output**

Tested it on Sample Test Cases



Also, Tested it on HackerRank Platform!



3.) Write a class to represent a vector (a series of float values). Include member functions to perform the following tasks:

* To **create** the vector
* To **modify the value** of a given element.
* To **multiply** by a scalar value.
* To **display the vector** in the form (10, 20, 30,…)

**Code**

*#include* <iostream>

*#include* <iomanip>

*#include* <assert.h>

*// U19CS012 [BHAGYA VINOD RANA]*

using namespace std;

*// Using Generic Class Programming, So it can be Used for Any Data Type {LL,Stack,Queue,..}*

template <class T>

class vector

{

    T \*p;

    int size;

public:

*// To create the vector*

    void create\_vector(T a);

    void set\_vector\_element(int i, T val);

*// To modify the value of a given element.*

    void modify\_vector(void);

*// To multiply by a scalar value.*

    void multiply\_vector(T b);

*// To display the vector in the form (10, 20, 30,…)*

    void display(void);

};

template <class T>

void vector<T>::create\_vector(T a)

{

    size = a;

    p = new float[size];

}

template <class T>

void vector<T>::set\_vector\_element(int i, T val)

{

    p[i] = val;

}

template <class T>

void vector<T>::multiply\_vector(T b)

{

*for* (int i = 0; i < size; i++)

        p[i] = b \* p[i];

}

template <class T>

void vector<T>::display(void)

{

    cout << "p[" << size << "] = ( ";

*for* (int i = 0; i < size; i++)

    {

*if* (i == size - 1)

            cout << p[i];

*else*

            cout << p[i] << " , ";

    }

    cout << ")" << endl;

}

template <class T>

void vector<T>::modify\_vector(void)

{

    cout << "~~~~> Task 2.1 - Modification by Insertion\n\n";

    int i;

    cout << "Enter Position of Element to be Deleted : ";

    cin >> i;

    assert(i >= 1 && i <= size);

    cout << "Enter the New Value of " << i << "th  element : ";

    T v;

    cin >> v;

*// 0 Based Indexing*

    i--;

    p[i] = v;

    cout << "New Vector Contents : " << endl;

    display();

    cout << "\n~~~~> Task 2.2 - Modification by Deletion\n\n";

    cout << "Enter Position of Element to be Deleted : ";

    cin >> i;

    assert(i >= 1 && i <= size);

    i--;

*for* (int j = i; j < size; j++)

    {

        p[j] = p[j + 1];

    }

    size--;

    cout << "New Vector Contents : " << endl;

    display();

}

int main()

{

    vector<float> v;

    cout << "\n~~~> Task 1 - Create Vector\n\n";

    int sz;

    cout << "Enter size of vector : ";

    cin >> sz;

    v.create\_vector(sz);

    cout << "Enter " << sz << " Elements {e1 e2 e3 .. en} :" << endl;

    float tmp;

*for* (int i = 0; i < sz; i++)

    {

        cin >> tmp;

        v.set\_vector\_element(i, tmp);

    }

    cout << "\n~~~> Task 4 - Display Vector\n\n";

    cout << "Vector Contents : " << endl;

    v.display();

    cout << "\n~~~> Task 3 - Multiply Vector with Scaler\n\n";

    cout << "Enter Scalar Float Number for Multiplication : ";

    cin >> tmp;

    v.multiply\_vector(tmp);

    cout << "Vector Contents : " << endl;

    v.display();

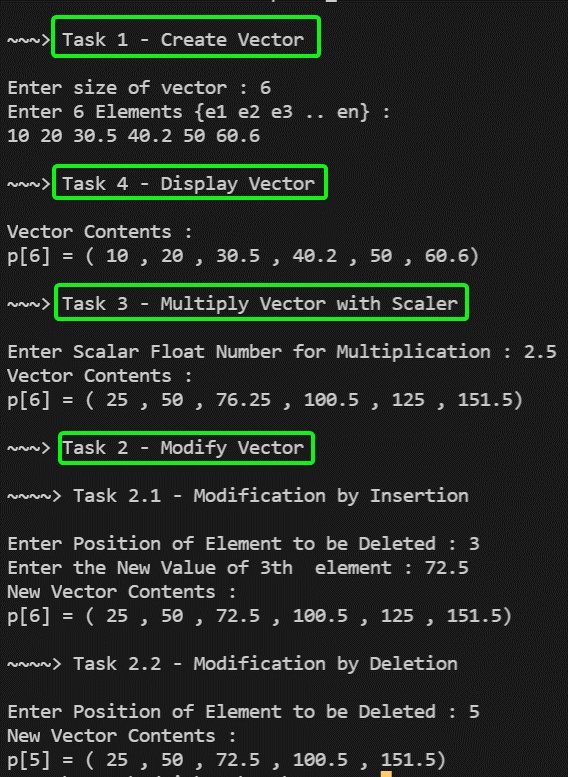
    cout << "\n~~~> Task 2 - Modify Vector\n\n";

    v.modify\_vector();

*return* 0;

}

**Output**



4.) A **book shop** maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position.

Whenever a customer wants a book, the sales person inputs **the title and author** and the system searches the list and displays whether it is available or not.

* If it is not, an appropriate message is displayed.
* If it is, then the system displays the **book details** and requests for the **number of copies** required.
* If the requested copies are **available**, the total cost of the requested copies is displayed; otherwise the message “Required copies not in stock” is displayed.

Design a system using a class called **books** with suitable member functions and constructors. Use **new** operators in constructors to allocate memory space required. Implement **C++ program** for the system.

Improve the system design to incorporate the following features:

* The **price of the books should be updated** as and when required. Use a member function to implement this.
* The **stock value of each book should be automatically** updated as soon as a transaction is completed.
* The number of **successful and unsuccessful transactions** should be recorded for the purpose of statistical analysis. Use **static data members** to keep count of transactions.
* Also demonstrate the use of pointers to access the members.

**Code**

*#include* <bits/stdc++.h>

using namespace std;

*// For Statistical Analysis of Transactions*

static int success = 0;

static int failure = 0;

*// Book Class*

class book

{

public:

    string author, title, publisher;

    int price, stock;

*// Book Constructor*

    book() {}

    book(string author, string title, string publisher, int price, int stock)

    {

*this*->author = author;

*this*->title = title;

*this*->publisher = publisher;

*this*->price = price;

*this*->stock = stock;

    }

*// To Check if Stock is Available or Not*

    bool is\_available()

    {

*return* stock > 0;

    }

*// Function to Check if Match is Found or Not*

    bool match(string title, string author)

    {

*return* *this*->title == title and *this*->author == author;

    }

*// Returns the Price if Stock is Available*

    float available(int copies)

    {

*if* (stock >= copies)

        {

            stock -= copies;

*return* (copies \* price);

        }

*else*

        {

*return* -1;

        }

    }

*// F(x) to Update the Price*

    void update\_price(int price)

    {

*this*->price = price;

    }

*// F(x) to Update the Stock*

    void update\_stock(int stock)

    {

*this*->stock += stock;

    }

};

*// Inventory Class*

class inventory

{

    vector<book \*> books;

public:

*// F(x) to add Book*

    void add\_book(book \*b)

    {

        books.push\_back(b);

        success++;

    }

*// F(x) to Search for Particular Book [Linear Search]*

    bool search\_book(string title, string author)

    {

*for* (int i = 0; i < books.size(); i++)

        {

*if* (books[i]->match(title, author))

            {

*return* true;

            }

        }

*return* false;

    }

*// F(x) to issue Book*

    bool issue\_book(string title, string author, int copies)

    {

*for* (int i = 0; i < books.size(); i++)

        {

*if* (books[i]->match(title, author))

            {

*if* (books[i]->is\_available())

                {

                    float cost = books[i]->available(copies);

*if* (cost != -1)

                    {

                        cout << "Book issued successfully. Cost : " << cost << endl;

                        success++;

*return* true;

                    }

*else*

                    {

                        cout << "Not Enough Copies Available. No Book Issued!" << endl;

                        failure++;

*return* false;

                    }

                }

*else*

                {

                    cout << "Book Not Available." << endl;

                    failure++;

*return* false;

                }

            }

        }

        failure++;

        cout << "Book Not Found." << endl;

*return* false;

    }

*// F(x) to Update the Price*

    void update\_price(string title, string author, int price)

    {

*for* (int i = 0; i < books.size(); i++)

        {

*if* (books[i]->match(title, author))

            {

                books[i]->update\_price(price);

                cout << "Price Updated." << endl;

                success++;

*return*;

            }

        }

        failure++;

        cout << "Book Not Found." << endl;

    }

*// F(x) to Update the Stock of Book*

    void update\_stock(string title, string author, int stock)

    {

*for* (int i = 0; i < books.size(); i++)

        {

*if* (books[i]->match(title, author))

            {

                books[i]->update\_stock(stock);

                cout << "Stock Updated." << endl;

                success++;

*return*;

            }

        }

        failure++;

        cout << "Book Not Found." << endl;

    }

};

*// F(x) to Print the Menu Options*

void menu()

{

    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

    cout << "1 -> Add Book" << endl;

    cout << "2 -> Search Book" << endl;

    cout << "3 -> Issue Book" << endl;

    cout << "4 -> Update Book Price" << endl;

    cout << "5 -> Update Book Stock" << endl;

    cout << "6 -> Statistical Analysis" << endl;

    cout << "7 -> Exit" << endl;

    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

}

int main()

{

    inventory store;

    book tmp\_book;

    string author, title, publisher;

    int price, stock, copies;

    int cost, choice = 0;

*while* (choice != 7)

    {

        menu();

        cout << "Enter your Choice : ";

        cin >> choice;

*switch* (choice)

        {

*case* 1:

        {

            cout << "Enter Author : ";

            cin >> author;

            cout << "Enter Title : ";

            cin >> title;

            cout << "Enter Publisher : ";

            cin >> publisher;

            cout << "Enter Price : ";

            cin >> price;

            cout << "Enter Stock : ";

            cin >> stock;

            tmp\_book = book(author, title, publisher, price, stock);

            store.add\_book(new book(tmp\_book));

            cout << "Book Added Successfully." << endl;

*break*;

        }

*case* 2:

        {

            cout << "Enter Author : ";

            cin >> author;

            cout << "Enter Title : ";

            cin >> title;

*if* (store.search\_book(title, author))

            {

                cout << "Book Found." << endl;

            }

*else*

            {

                cout << "Book Not Found." << endl;

            }

*break*;

        }

*case* 3:

        {

            cout << "Enter Author : ";

            cin >> author;

            cout << "Enter Title : ";

            cin >> title;

            cout << "Enter No. of Copies: ";

            cin >> copies;

            store.issue\_book(title, author, copies);

*break*;

        }

*case* 4:

        {

            cout << "Enter Author : ";

            cin >> author;

            cout << "Enter Title : ";

            cin >> title;

            cout << "Enter New Price : ";

            cin >> price;

            store.update\_price(title, author, price);

*break*;

        }

*case* 5:

        {

            cout << "Enter Author : ";

            cin >> author;

            cout << "Enter Title : ";

            cin >> title;

            cout << "Enter Stock to be Added : ";

            cin >> stock;

            store.update\_stock(title, author, stock);

*break*;

        }

*case* 6:

        {

            cout << "Successful Transactions : " << success << endl;

            cout << "Failure Transactions    : " << failure << endl;

*break*;

        }

*case* 7:

        {

            cout << "Thank You for Visiting Our Book Shop!" << endl;

*break*;

        }

*default*:

        {

            cout << "Invalid Choice Entered." << endl;

*break*;

        }

        }

    }

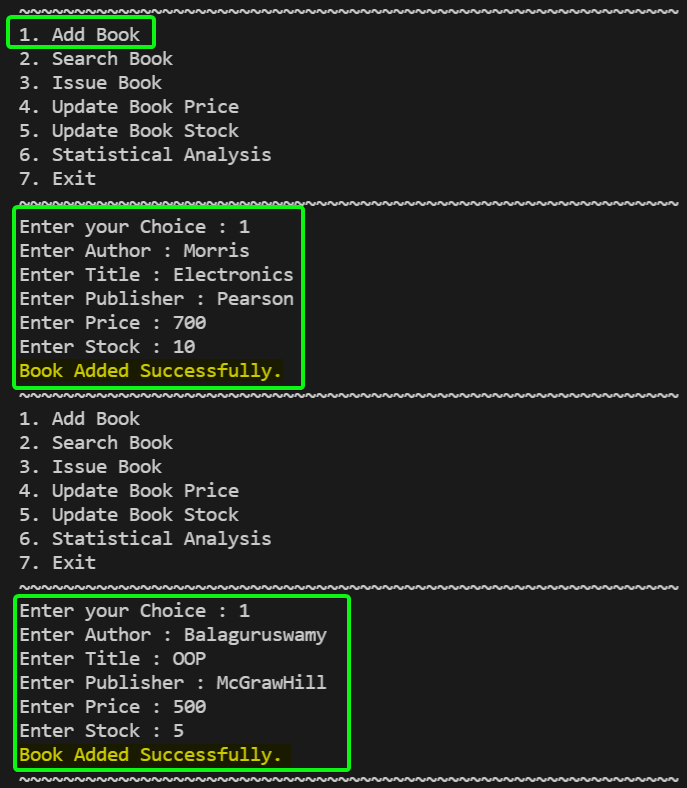
    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

*return* 0;

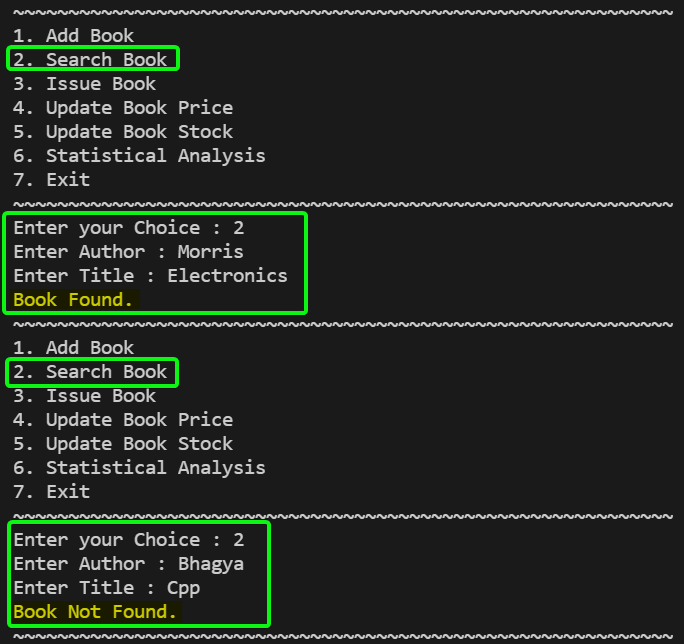
}

**Output**

Task 1: Adding a Book



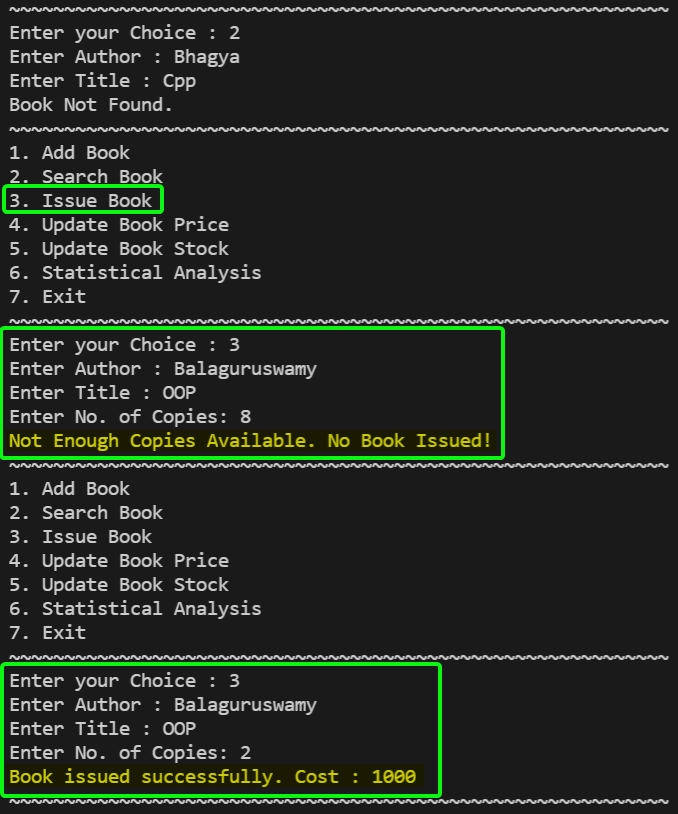
Task 2: Searching a Book



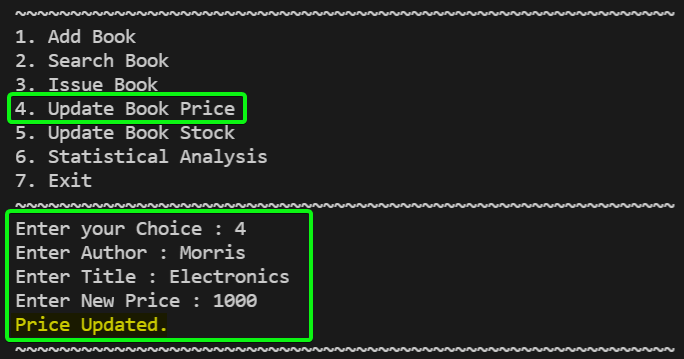
Task 3: Issuing a Book

We will first Try to Issue a Book with Larger than Stock Available.

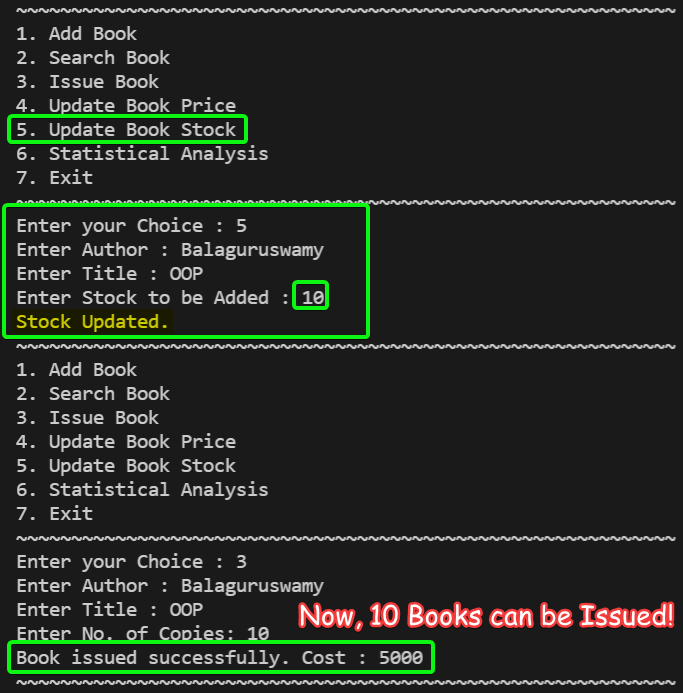
Secondly, We will Check with Quantity within Stock Available and get the Cost after issuing the Book.



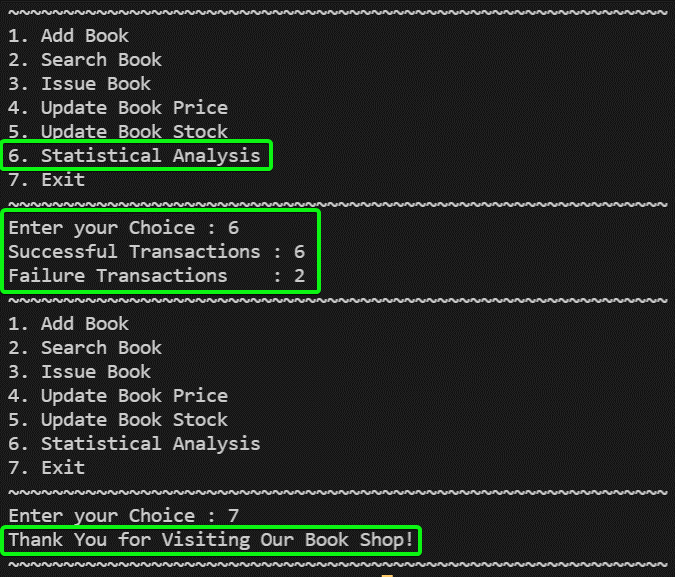
Task 4: Updating Book Price



Task 5: Updating Stocks



Task 6: Statistical Analysis



**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA