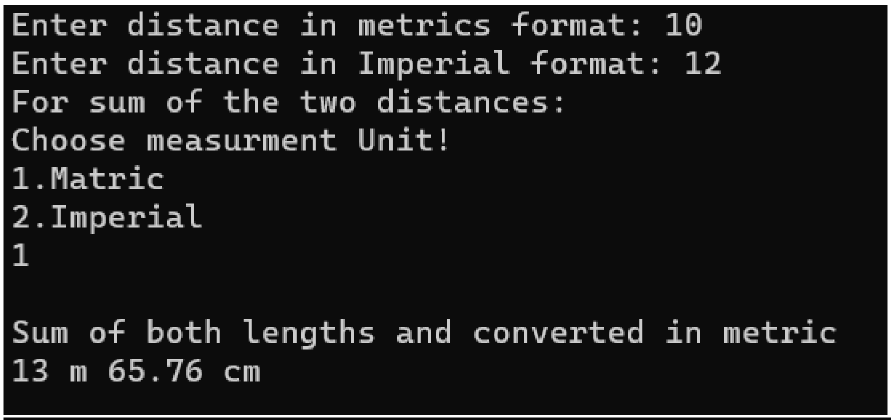
Assignment No: 2 Date:

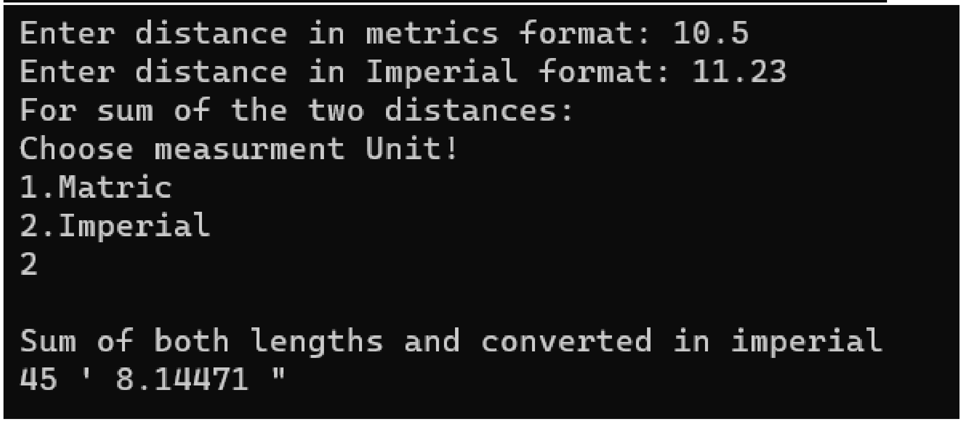
A] Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimetres and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB.
Use a friend function to carry out the addition operation. The object that stores the results may be a DM object or DB object, depending on the units in which the results are required.
The display should be in the format of feet and inches or metres and centimetres depending on the object on display.

```
#include<iostream>

using namespace std;
class DB;
class DM{
   int meter;
   float centimeter;
public:
   DM(){          //metric units
      meter = 0;
      centimeter = 0;
   }
   DM(float d){
      meter = d;
      centimeter = 0;
      centimeter =(d - meter) * 100;
   }
   friend DM sum(DM&, DB&);
   friend DB sum(DB&, DM&);
   void display(){
      cout<<meter<<" m "<<centimeter<<" cm"<<endl;
   }
};

class DB{          //imperial units
   int feet;
   float inches;
public:
   DB(){
      feet = 0;
      inches = 0;
   }
   DB(float d){
      feet = d;
      inches = 0;
      inches = (d - feet) * 12;
   }
   friend DM sum(DM&, DB&);
```

```
    friend DB sum(DB&, DM&);
    void display(){
        cout<<feet<<" \' "<<inches<<" \""<<endl;
    }
};

DB sum(DB &b, DM &a){
    float distance = b.feet + b.inches * 0.0833 + (a.meter + (a.centimeter)/100)* 3.28084;
    DB c(distance);
    return c;
}
DM sum(DM &a, DB &b){
    float distance = a.meter + (a.centimeter / 100) + ((b.feet + (b.inches * 0.0833)) * 0.3048);
    DM c(distance);
    return c;
}

int main()
{
    float distance;

    cout<<"Enter distance in metrics format: ";cin>>distance;
    DM m_dist(distance);
    cout<<"Enter distance in Imperial format: ";cin>>distance;
    DB i_dist = DB(distance);

    cout<<"For sum of the two distances: "<<endl;
    int op;
    cout<<"Choose measurment Unit!"<<endl;
    cout<<"1.Matric"<<endl<<"2.Imperial"<<endl;
    cin>>op;

    if(op == 1){
        cout<<"\nSum of both lengths and converted in metric"<<endl;
        m_dist = sum(m_dist, i_dist);
        m_dist.display();
    }
    else if(op == 2){
        cout<<"\nSum of both lengths and converted in imperial"<<endl;
        i_dist = sum(i_dist, m_dist);
        i_dist.display();
    }

    return 0;
}
```



```
Enter distance in metrics format: 10
Enter distance in Imperial format: 12
For sum of the two distances:
Choose measurment Unit!
1.Matric
2.Imperial
1

Sum of both lengths and converted in metric
13 m 65.76 cm
```



```
Enter distance in metrics format: 10.5
Enter distance in Imperial format: 11.23
For sum of the two distances:
Choose measurment Unit!
1.Matric
2.Imperial
2

Sum of both lengths and converted in imperial
45 ' 8.14471 "
```

B] Write a class to represent a vector (a series of float values). Include member functions to perform the following tasks:
(a) To create the vector
(b) To modify the value of a given element
(c) To multiply by a scalar value
(d) To display the vector in the form (10, 20, 30, ...)
Write a program to test your class.
Modify the class and program such that the program would be able to add two vectors and display the resultant vector. (Note that we can pass objects as function arguments.)

```
#include <iostream>

using namespace std;

class vector{
  float *ptr;
  int size;
public:
  vector(){
    ptr = NULL;
  }
  vector(int n){
    ptr = new float [n];
    size = n;
  }
  ~vector(){
    delete(ptr) ;
    }
    void create_vector(){
        for(int i = 0; i<size; i++){
            cout<<"\nEnter Element "<<i+1<<" : "; cin>>ptr[i];
        }
    }
    bool modify(int pos, float num){
        if(pos > size){
            return false;
        }
        ptr[pos-1] = num;
        return true;
    }
    friend vector multipy_byconst(vector &vec, float c);
    friend vector add_vectors(vector a, vector b);
    void display();
};

vector multipy_byconst(vector &vec, float c){
    if(vec.ptr != NULL)
        for(int i=0; i<vec.size; i++){
            vec.ptr[i] *= c;
        }
    else
        cout<<"\nEmpty vector"<<endl;
    return vec;
}
vector add_vectors(vector a, vector b){
    int size = a.size>b.size ? a.size:b.size;
    vector c(size);
    for(int i=0; i<size; i++){
            c.ptr[i] = a.ptr[i] + b.ptr[i];
    }
    return c;
}
void vector :: display(){
    if(ptr != NULL)
        for(int i=0; i<size; i++){
            cout<<ptr[i];
            if(i != size-1)
                cout<<" ,";
    }
    else
        cout<<"\nEmpty vector"<<endl;
}

int main()
{
    int size,op, pos;
```

```
cout<<"Enter size of vector A : "; cin>>size;
vector A(size);
cout<<"vector A : " ; A.create_vector();
cout<<"Enter size of vector B : "; cin>>size;
vector B(size);
cout<<"vector B : "; B.create_vector();

cout<<"Vecotr A : "; A.display(); cout<<endl;
cout<<"Vector B : "; B.display(); cout<<endl;

cout<<"Vector to modify\n1.Vector A\n2.Vector B"<<endl;
cin>>op;
cout<<"Enter position: "; cin>>pos;
int element;
   cout<<"Enter element to modify: ";cin>>element;
if(op==1){
   if(A.modify(pos, element))
      cout<<"Element added sucessfully !"<<endl;
   else cout<<"Position not found !"<<endl;
}
else if(op == 2){
   if(B.modify(pos, element))
      cout<<"Element added sucessfully !"<<endl;
   else cout<<"Position not found !"<<endl;
}
else cout<<"Error! vector not found"<<endl;

cout<<"Vecotr A : "; A.display(); cout<<endl;
cout<<"Vector B : "; B.display(); cout<<endl;

float tmp;
cout<<"Multiply by constant\n1. Vector A\n2.Vector B"<<endl;cin>>op;
cout<<"Enter constant to multiply: "; cin>>tmp;
if(op == 1){
   A = multipy_byconst(A,tmp);
}
else if(op == 2){
   B = multipy_byconst(B,tmp);
}
else cout<<"Error! vector not found"<<endl;

cout<<"Vecotr A : "; A.display(); cout<<endl;
cout<<"Vector B : "; B.display(); cout<<endl;

cout<<"Add two vectors to\n1. Vector A\n2.Vector B"<<endl;cin>>op;
if(op == 1){
   A = add_vectors(A,B);
}
```

```
    else if(op == 2){
        B = add_vectors(B,A);
    }
    else cout<<"Error! vector not found"<<endl;

    cout<<"Vecotr A : "; A.display(); cout<<endl;
    cout<<"Vector B : "; B.display(); cout<<endl;

    return 0;
}
```

C] Describe the mechanism of accessing data members and member functions in the following cases:
(a) Inside the main program.
(b) Inside a member function of the same class.
(c) Inside a member function of another class

(a) Inside the main program:
In this case, you're working with an instance of a class from the main program. To access data members and member functions:

Data Members:

Use the dot (.) operator to access data members. For example, if you have a class MyClass with a data member my_variable, you would access it like this:

```
MyClass obj;
obj.my_variable = 10; // Assigning a value to my_variable
```

Member Functions:

Use the dot (.) operator to call member functions. For example, if you have a member function my_function() in MyClass, you would call it like this:

```
MyClass obj;
obj.my_function(); // Calling my_function
```

(b) Inside a member function of the same class:
In this case, you're already inside a member function of the class. To access data members and member functions:

Data Members:

Since you're already inside the class, you can directly access data members without using any operator. For example:

```
class MyClass {
    int my_variable; // Data member
    void my_function() {
        my_variable = 10; // Accessing my_variable directly
    }
};
```

Member Functions:

Similarly, you can call other member functions directly:

```
class MyClass {
   void function1() {
      // ...
   }
   void function2() {
      function1(); // Calling another member function
   }
};
```

(c) Inside a member function of another class:
In this case, you're inside a member function of one class and want to access data members and member functions of another class.

Data Members:

If you have an instance of the other class, you can use the dot (.) operator to access its data members:

```
class Class1 {
public:
   int data;
};

class Class2 {
public:
   void my_function(Class1 &obj) {
      obj.data = 10; // Accessing data member of Class1
   }
};
```

Member Functions:

Similarly, you can call member functions of the other class using the dot (.) operator:

```
class Class1 {
public:
   void my_function() {
      // ...
   }
};

class Class2 {
public:
   void my_other_function(Class1 &obj) {
      obj.my_function(); // Calling a member function of Class1
   }
};
```

D] 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 operator in constructors to allocate memory space required.
(a) The price of the books should be updated as and when required. Use a private member function to implement this.
(b) The stock value of each book should be automatically updated as soon as a transaction is completed.
(c) 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.

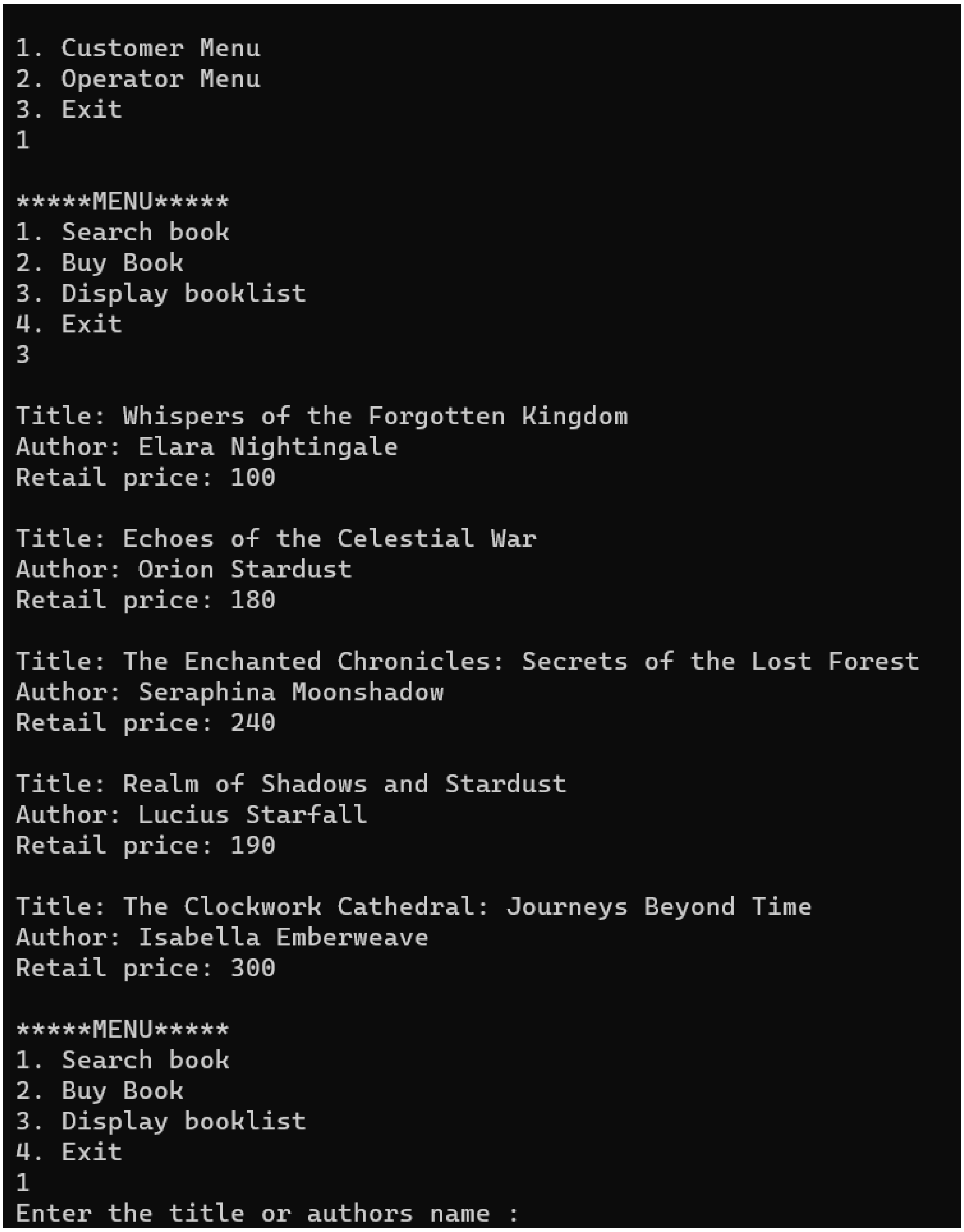
```
#include<iostream>
#include<stdlib.h>
#include<string.h>
#include <fstream>
using namespace std;

class book
{
   string title;
   string author;
   float retail_price;
   string publisher;
   int stock;
   void edit_price(float _price){
      retail_price = _price;
   }
public:
   book *next,* prev;
   book()
   {
      cout<<"Enter Title: "; getline(cin, title, '\n');
      cout<<"Enter author: "; getline(cin, author, '\n');
      cout<<"Enter retail price: "; cin>>retail_price; getchar();
      cout<<"Enter publisher: "; getline(cin, publisher, '\n');
      cout<<"Enter book stock: "; cin>>stock; getchar();
      next = prev = NULL;
   }
   string get_author(){
      return author;
   }
   string get_title(){
      return title;
   }
   int find_(string _title, string _author){
        return ((title == _title || author == _author));
    }
    int availablity(){
        return stock;
    }
    int edit_stock(int _stock){
        stock += _stock;
        return stock;
    }
    void display(){
        cout<<"Title: "<<title<<endl;
        cout<<"Author: "<<author<<endl;
        cout<<"Retail price: "<<retail_price<<endl;
        cout<<"Publisher: "<<publisher<<endl;
        cout<<"Stock: "<<stock<<endl;
    }
    void display_for_list(){
        cout<<"\nTitle: "<<title<<endl;
        cout<<"Author: "<<author<<endl;
        cout<<"Retail price: "<<retail_price<<endl;
    }

    // for file loading

    book(string t,string a,float p, string pup, int s){
        title = t;
        author = a;
        retail_price = p;
        publisher = pup;
        stock = s;
        prev = next = NULL;
    }

};


class booklist{
    book *head;
public:
    booklist(){
        head = NULL;
    }
    void load_stock();
    void display_booklist();//
    book *search_book();//
    void buy_book();//
    void new_stock();//
    void delete_book();//
```

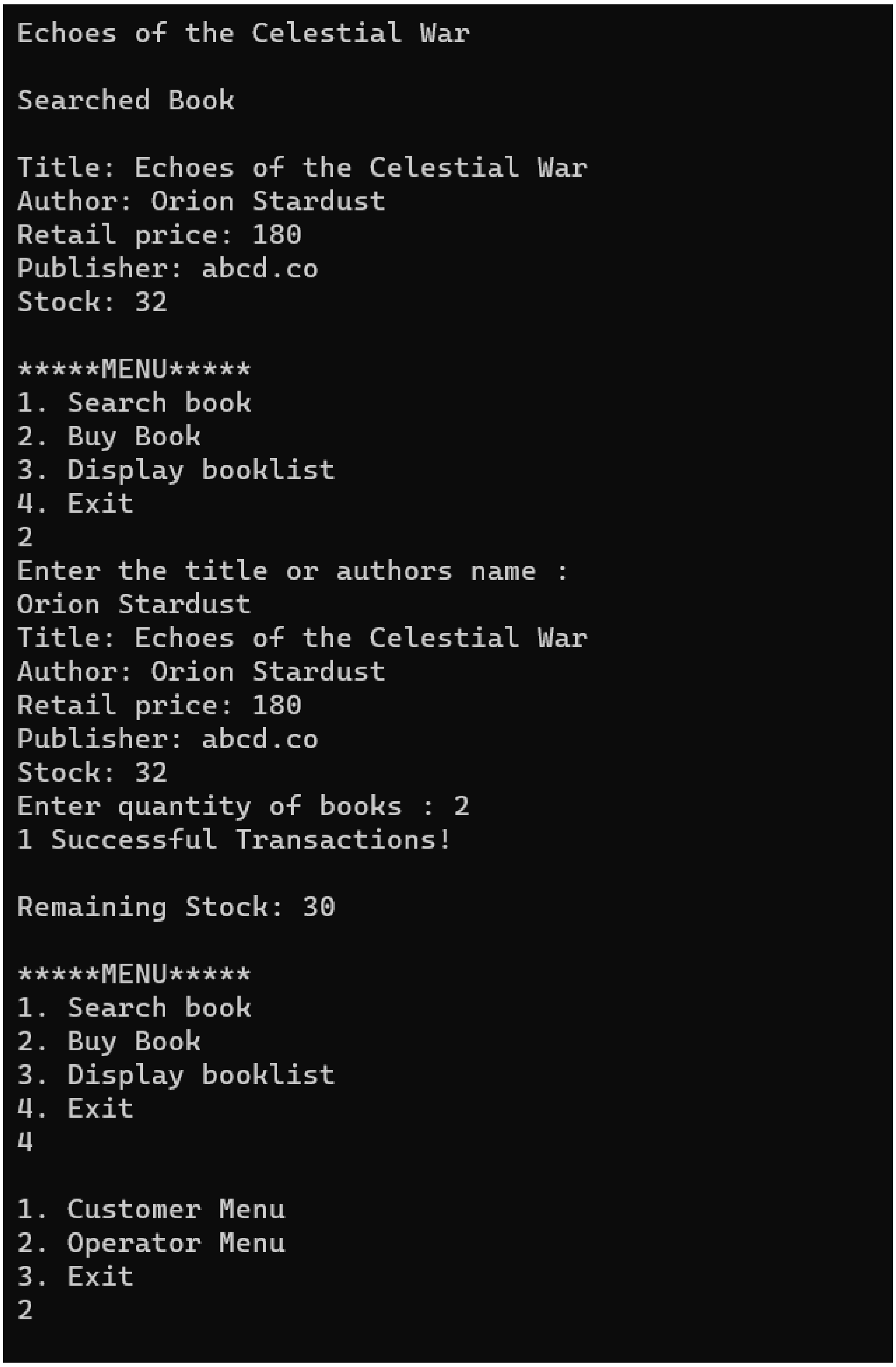
```
};

void booklist :: new_stock()
{
    book * tmp = new book;
    if(head == NULL){
        head = tmp;
        tmp->prev = tmp->next = NULL;
        return;
    }
    book *p = head, *prev;
    while(p != NULL){
        if(p->find_(tmp->get_title(),tmp->get_author())){
            cout<<"Book already exists!"<<endl;
            return;
        }
        if(p->next == NULL){
            p->next = tmp;
            tmp->prev = p;
            tmp->next = NULL;
            return;
        }
        prev = p;
        p = p->next;
    }
}

book* booklist :: search_book()
{
    string term;
    cout<<"Enter the title or authors name : "<<endl;
    getline(cin, term, '\n');
    book *p = head;
    while(p != NULL){
        if(p->find_(term,term)){
            break;
        }
        p = p->next;
    }
    return p;
}

void booklist :: buy_book()
{
    static int s=0, us=0;
    book* tmp;
    tmp = search_book();
    if(tmp == NULL){
        us ++;
        cout<<"Book not available!"<<endl;
        cout<<us<<" Unsuccsessful transactions!"<<endl;
        return;
    }
    else {
        int q;
        tmp->display();
        cout<<"Enter quantity of books : ";
        cin>>q;
        if(tmp->availablity() >= q){
            s ++;
            tmp->edit_stock(-q);
            cout<<s<<" Successful Transactions!"<<endl;
            cout<<"\nRemaining Stock: "<< tmp->availablity()<<endl;
        }
        else{
            cout<<"Not sufficient stock!\nTry again!"<<endl;
        }
    }
}

void booklist :: delete_book()
{
    book* tmp;
    tmp = search_book();
    if(tmp == NULL){
        cout<<"Book not available!"<<endl;
        return;
    }
    else {
        tmp->display();
        cout<<"\nDeleting Stock...\n"<<endl;
        if(head->next == NULL){
            cout<<"\nBook Store is now Empty"<<endl;
        }else if(tmp == head){
            head = tmp->next;
            tmp->next->prev = NULL;
        }else if(tmp->next == NULL){
            tmp->prev->next = NULL;

        }else{
            tmp->prev->next = tmp->next;
            tmp->next->prev = tmp->prev;
        }
        delete tmp;
    }
}
```

```
void booklist :: display_booklist()
{
   book* p;
   if(head == NULL){
      cout<<"\nNo books available!\n"<<endl;
      return;
   }
   p = head;
   while(p != NULL){
      p->display_for_list();
      p = p->next;
   }
}

void booklist :: load_stock()
{
   string title, author,pr, publisher, st;
   int stock;
   float price;
   int count = 0;
   ifstream myfile ("booklist.txt");
   if (myfile.is_open())
   {
      book* p;
      p = NULL;
      while ( getline (myfile,title,'\n') )
      {
         getline (myfile,author,'\n');
         getline (myfile,pr,'\n');
         price = stof(pr);
         getline (myfile,publisher,'\n');
         getline (myfile,st,'\n');
         stock = stof(st);
         book* tmp = new book(title,author,price,publisher,stock);
         count ++;
         if(p == NULL)
            head = tmp;
         else{
            p->next = tmp;
            tmp->prev = p;
         }
         p = tmp;
      }
   }
   else cout << "Unable to open file";
   myfile.close();
}
```

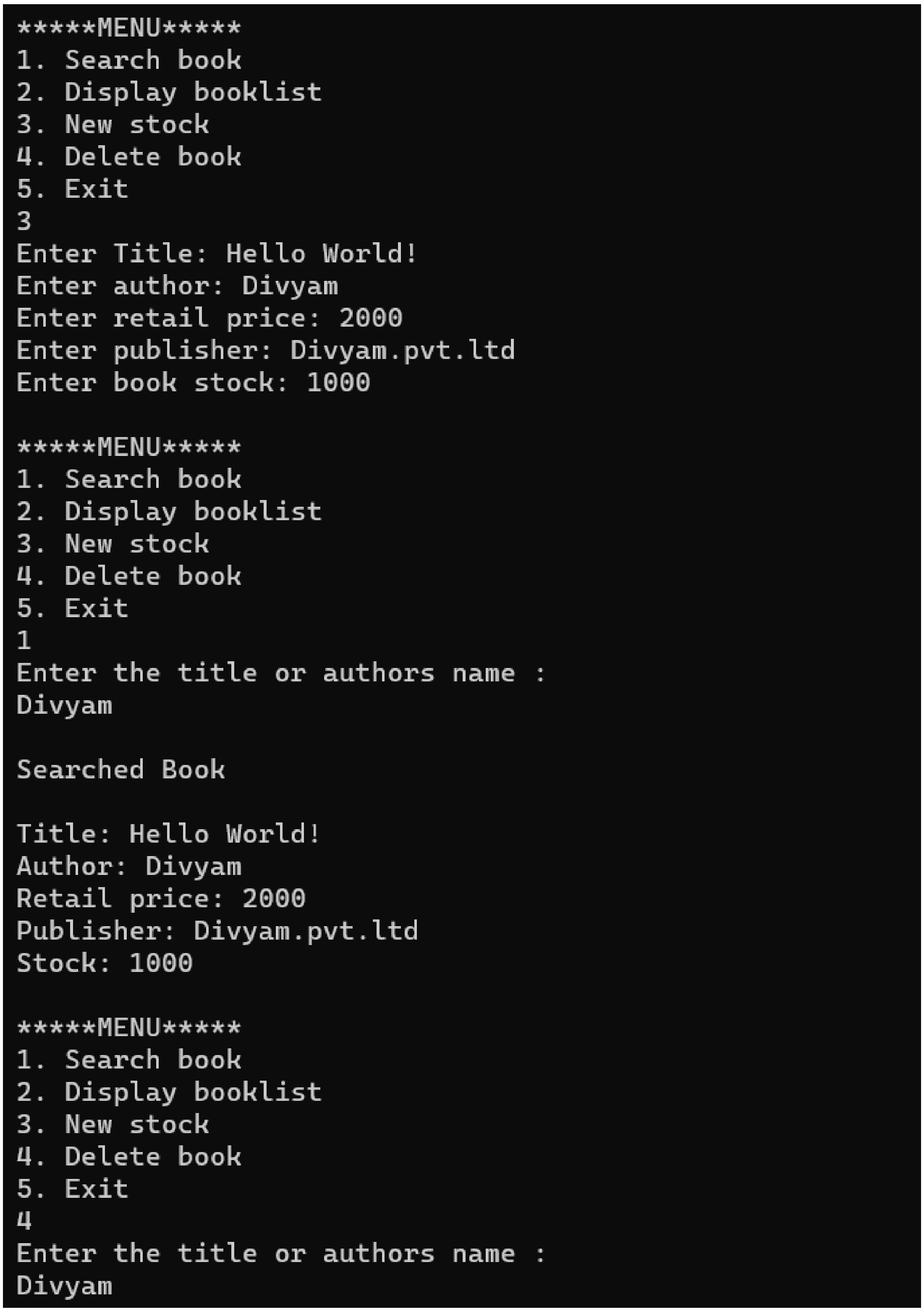
```
void MENU_for_operator(booklist & bk_list)
{
   int op;
   book* bk;
   while(true){
   cout<<"\n*****MENU*****\n";
   cout<<"1. Search book \n2. Display booklist\n3. New stock\n4. Delete book\n5. Exit"<<endl;
   cin>>op; getchar();

   switch(op){
   case 1: if((bk = bk_list.search_book()) == NULL){
              cout<<"Book Not Found!"<<endl;
           }else{
              cout<<"\nSearched Book\n"<<endl;
              bk->display();
           }
           break;
   case 2: bk_list.display_booklist(); break;
   case 3: bk_list.new_stock(); break;
   case 4: bk_list.delete_book(); break;
   case 5: return;
   default : cout<<"wrong Option Entered!"<<endl; break;
   }
   }
}


void MENU_for_customr(booklist & bk_list)
{
   int op;
   book* bk;
   while(true){
   cout<<"\n*****MENU*****\n";
   cout<<"1. Search book \n2. Buy Book\n3. Display booklist\n4. Exit"<<endl;
   cin>>op; getchar();
   switch(op){
   case 1: if((bk = bk_list.search_book()) == NULL){
              cout<<"Book Not Found!"<<endl;
           }else{
              cout<<"\nSearched Book\n"<<endl;
              bk->display();
           }
           break;
   case 2: bk_list.buy_book(); break;
   case 3: bk_list.display_booklist();break;
   case 4: return;
   default : cout<<"wrong Option Entered!"<<endl; break;
```

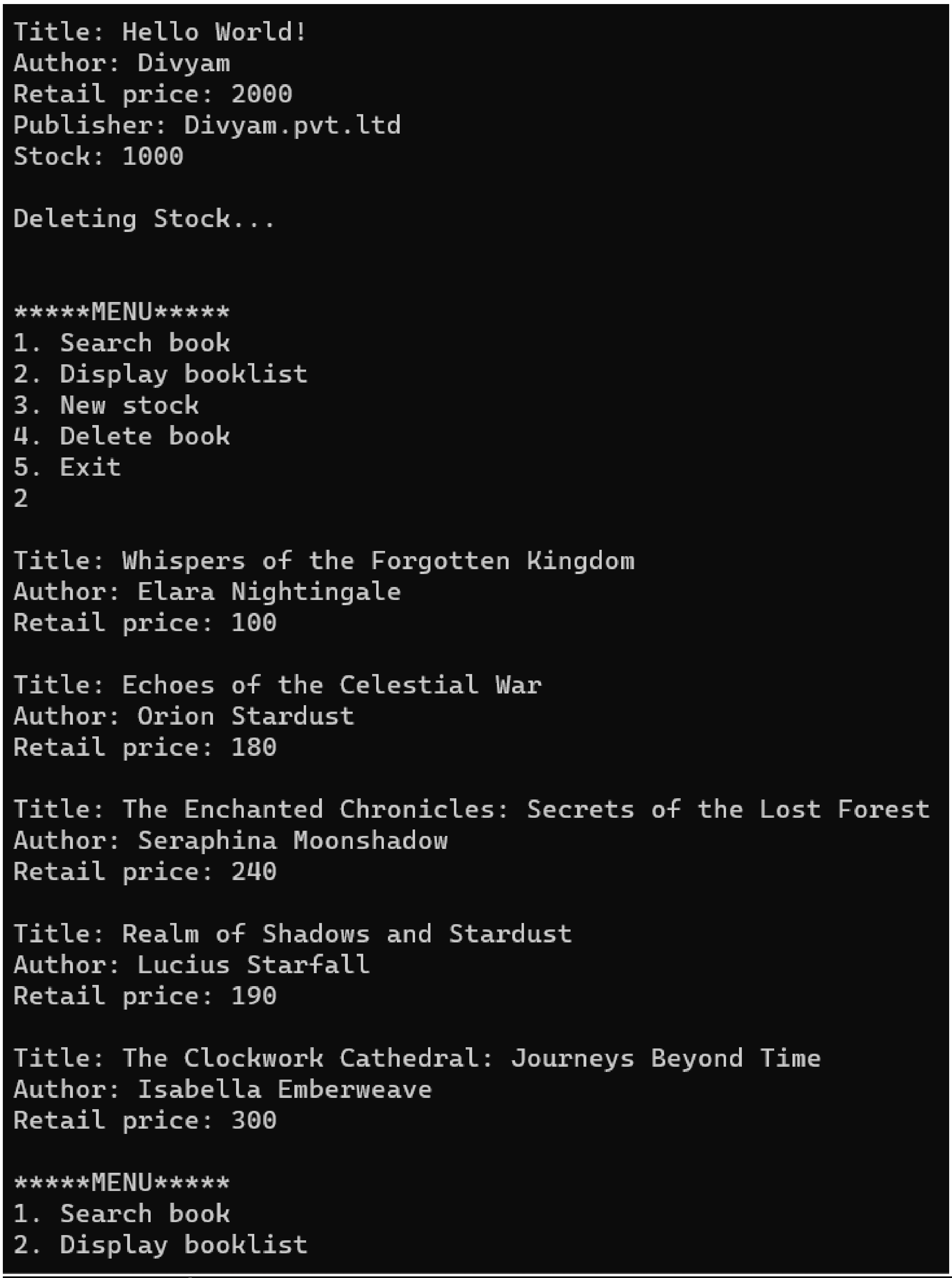
```
    }
    }
}

int main()
{
    int op;
    booklist bk_list;
    bk_list.load_stock();
    while(true){
        cout<<"\n1. Customer Menu\n2. Operator Menu\n3. Exit"<<endl;
        cin>>op;
        if(op == 1){
            MENU_for_customr(bk_list);
        }else if(op == 2){
            MENU_for_operator(bk_list);
        }else if(op == 3){return 0;}
        else cout<<"wrong Option Entered"<<endl;
    }
    return 0;
}
```


```
1. Customer Menu
2. Operator Menu
3. Exit
1

*****MENU*****
1. Search book
2. Buy Book
3. Display booklist
4. Exit
3

Title: Whispers of the Forgotten Kingdom
Author: Elara Nightingale
Retail price: 100

Title: Echoes of the Celestial War
Author: Orion Stardust
Retail price: 180

Title: The Enchanted Chronicles: Secrets of the Lost Forest
Author: Seraphina Moonshadow
Retail price: 240

Title: Realm of Shadows and Stardust
Author: Lucius Starfall
Retail price: 190

Title: The Clockwork Cathedral: Journeys Beyond Time
Author: Isabella Emberweave
Retail price: 300

*****MENU*****
1. Search book
2. Buy Book
3. Display booklist
4. Exit
1
Enter the title or authors name :
```

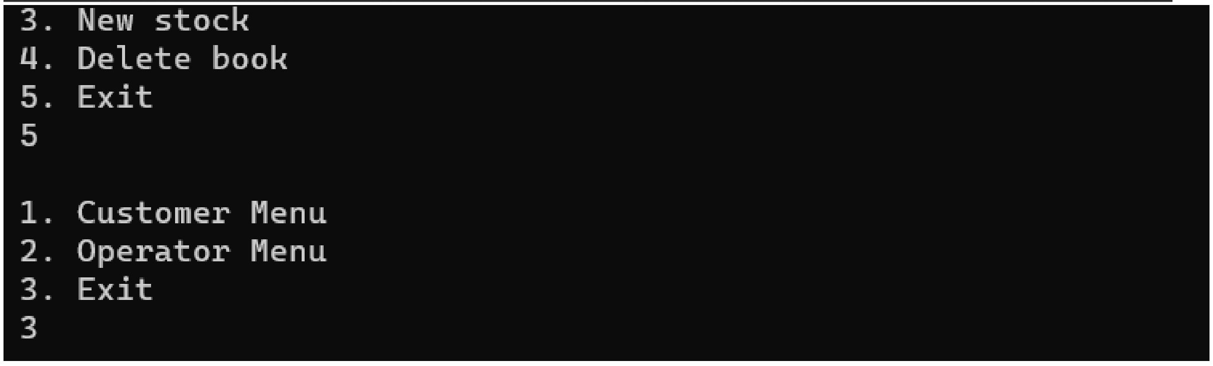


```
Echoes of the Celestial War

Searched Book

Title: Echoes of the Celestial War
Author: Orion Stardust
Retail price: 180
Publisher: abcd.co
Stock: 32

*****MENU*****
1. Search book
2. Buy Book
3. Display booklist
4. Exit
2
Enter the title or authors name :
Orion Stardust
Title: Echoes of the Celestial War
Author: Orion Stardust
Retail price: 180
Publisher: abcd.co
Stock: 32
Enter quantity of books : 2
1 Successful Transactions!

Remaining Stock: 30

*****MENU*****
1. Search book
2. Buy Book
3. Display booklist
4. Exit
4

1. Customer Menu
2. Operator Menu
3. Exit
2
```


```
*****MENU*****
1. Search book
2. Display booklist
3. New stock
4. Delete book
5. Exit
3
Enter Title: Hello World!
Enter author: Divyam
Enter retail price: 2000
Enter publisher: Divyam.pvt.ltd
Enter book stock: 1000

*****MENU*****
1. Search book
2. Display booklist
3. New stock
4. Delete book
5. Exit
1
Enter the title or authors name :
Divyam

Searched Book

Title: Hello World!
Author: Divyam
Retail price: 2000
Publisher: Divyam.pvt.ltd
Stock: 1000

*****MENU*****
1. Search book
2. Display booklist
3. New stock
4. Delete book
5. Exit
4
Enter the title or authors name :
Divyam
```



```
Title: Hello World!
Author: Divyam
Retail price: 2000
Publisher: Divyam.pvt.ltd
Stock: 1000

Deleting Stock...


*****MENU*****
1. Search book
2. Display booklist
3. New stock
4. Delete book
5. Exit
2

Title: Whispers of the Forgotten Kingdom
Author: Elara Nightingale
Retail price: 100

Title: Echoes of the Celestial War
Author: Orion Stardust
Retail price: 180

Title: The Enchanted Chronicles: Secrets of the Lost Forest
Author: Seraphina Moonshadow
Retail price: 240

Title: Realm of Shadows and Stardust
Author: Lucius Starfall
Retail price: 190

Title: The Clockwork Cathedral: Journeys Beyond Time
Author: Isabella Emberweave
Retail price: 300

*****MENU*****
1. Search book
2. Display booklist
```



```
3. New stock
4. Delete book
5. Exit
5

1. Customer Menu
2. Operator Menu
3. Exit
3
```

Nitesh Naik

(Subject Faculty)