BHARATIVIDYAPEETH'SCOLLEGEOFENGINEERING
(Approved by AICTE, New Delhi & Affiliated to Guru Gobind Singh Indraprastha University, Delhi)
(AnISO9001:2015CertifiedInstitution)
A-4, Paschim Vihar, Main Rohtak Road, New Delhi– 110 063

DEPARTMENT OF INFORMATION TECHNOLOGY

Project Based Learning

PBL Problem Statement: Online bookstore using binary search trees

Course Name: Data Structures

Student Name: Ayush Kumar Singh

E. No.: 02711503123

MAXIMUM MARKS: 05
(To be filled by faculty member)

Criteria	Achieved (✓)	Not Achieved (X)	Marks
Knowledge (Remember)			
Comprehension (Understand)			
Application (Apply)			
Analysis (Analyze)			
Synthesis (Create)			
Evaluation (Evaluate)			

Faculty Name & Signature with date

AIM:- ONLINE BOOKSTORE USING BINARY SEARCH TREE

We have to implement an book online book store management system using binary search trees, file organization, hash maps, and arrays considering the admin perspective, and implement ADT of binary search trees.

About The Project :-

An online bookstore management system for the business owner to perform the following functions. generation of bill, adding new book titles, deleting book titles, updating the quantity of books and checking the availability of books.

To generate the bill, we have taken customer details and stored them in an array and then we take input of books and the bill is displayed and the quantity from the stock is reduced. The purchase details and the total amount to be paid is then displayed in a bill.

For adding a book in stock, the book name and its respective price is entered by the user and then with the help of the tree insert function and hashmap, the new book data is added to the book list. Simultaneously

the array index for the stock array is incremented to keep a track of the number of new books added.

For deleting a particular book, the book name is entered by the user. If the book name entered matches the book data present in the tree, then the book is directly deleted by using the tree. The remove key function deletes from the hashmap as well.

For updating the current book stock, the book name is taken as user input, if the book is present in the tree, then the user is asked to enter the required quantity of books. Then the stock is updated by adding the new book in the array.

All the book details entries are stored in the hashmap by using a set of entries. Then the values are printed using the entry.

DELTA Pg No.
Date / /

Data Structures used :-

Binary search trees to store the books in a heirarchial way and sorted order. Make traversal, insertion and deletion faster. No size limit and can store ~~books~~ as many nodes as possible. Code is comparitovily simple.

Array is used to store quantity and price of the books and another array to access book details for generation of bill. Can store multiple data of similar types. Most familiar data structure.

Hash maps stores the book name and index in key: value pair. One object is used as key ~~and~~ (index) to another object (value). Faster access of elements restricts duplicate keys. Adding and removing elements from a hashmap based on a key takes constant time.

File organization is used to read data from text files and return in byte format.

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_BOOKS 100
#define MAX_NAME 50

typedef struct Book {
    char title[MAX_NAME];
    int quantity;
    double price;
    struct Book *left, *right;
} Book;

Book *root = NULL;
int bookIndex = 0;
char titles[MAX_BOOKS][MAX_NAME];
double prices[MAX_BOOKS];
int quantities[MAX_BOOKS];

Book *createBook(char *title, double price, int quantity) {
    Book *newBook = (Book *)malloc(sizeof(Book));
    strcpy(newBook->title, title);
    newBook->price = price;
    newBook->quantity = quantity;
    newBook->left = newBook->right = NULL;
    return newBook;
}

Book *insertBook(Book *node, char *title, double price, int quantity) {
    if (!node) return createBook(title, price, quantity);
    if (strcmp(title, node->title) < 0)
        node->left = insertBook(node->left, title, price, quantity);
    else if (strcmp(title, node->title) > 0)
        node->right = insertBook(node->right, title, price, quantity);
    return node;
}

Book *findMin(Book *node) {
    while (node->left) node = node->left;
    return node;
}

Book *deleteBook(Book *node, char *title) {
    if (!node) return node;
    if (strcmp(title, node->title) < 0)
        node->left = deleteBook(node->left, title);
    else if (strcmp(title, node->title) > 0)
        node->right = deleteBook(node->right, title);
    else {
```

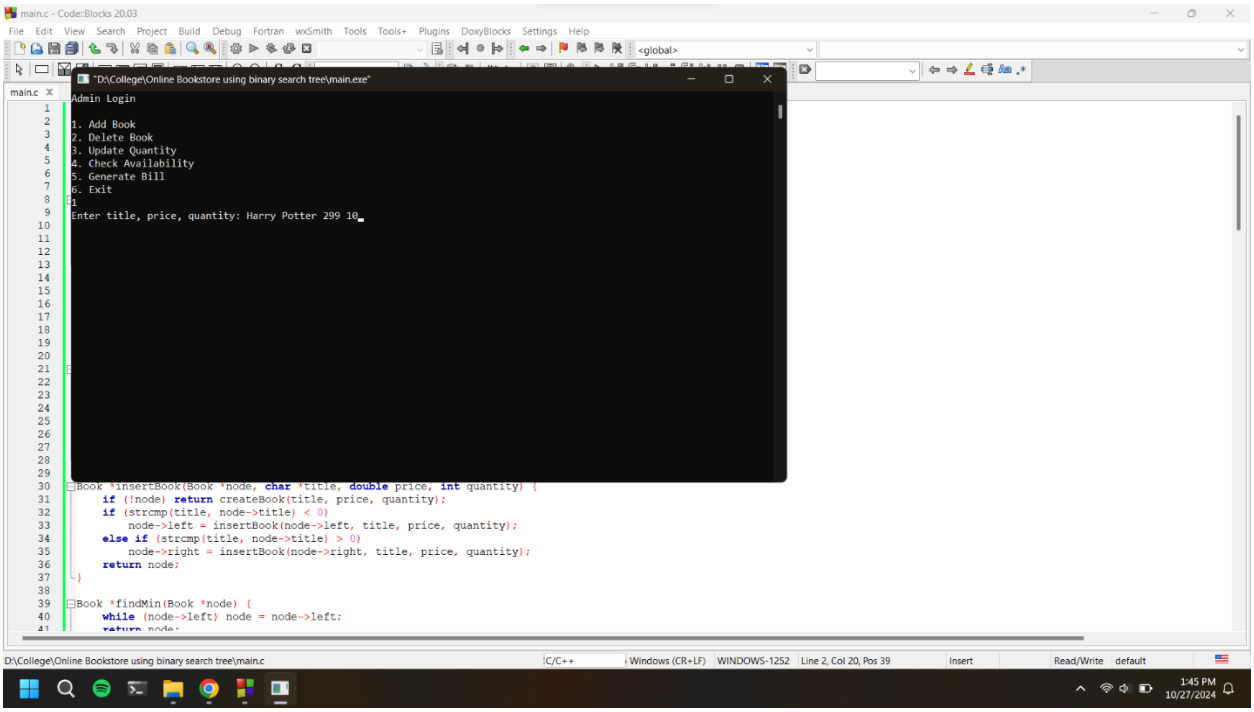
```
        if (!node->left) {
            Book *temp = node->right;
            free(node);
            return temp;
        } else if (!node->right) {
            Book *temp = node->left;
            free(node);
            return temp;
        }
        Book *temp = findMin(node->right);
        strcpy(node->title, temp->title);
        node->price = temp->price;
        node->quantity = temp->quantity;
        node->right = deleteBook(node->right, temp->title);
    }
    return node;
}

Book *searchBook(Book *node, char *title) {
    if (!node || strcmp(title, node->title) == 0) return node;
    if (strcmp(title, node->title) < 0)
        return searchBook(node->left, title);
    return searchBook(node->right, title);
}

void addBook(char *title, double price, int quantity) {
    root = insertBook(root, title, price, quantity);
    strcpy(titles[bookIndex], title);
    prices[bookIndex] = price;
    quantities[bookIndex++] = quantity;
}

void updateQuantity(char *title, int newQuantity) {
    Book *book = searchBook(root, title);
    if (book) {
        book->quantity = newQuantity;
        for (int i = 0; i < bookIndex; i++) {
            if (strcmp(titles[i], title) == 0) {
                quantities[i] = newQuantity;
                break;
            }
        }
    }
}

void checkAvailability(char *title) {
    Book *book = searchBook(root, title);
    if (book) printf("Title: %s, Price: %.2f, Quantity: %d\n", book->title,
book->price, book->quantity);
    else printf("Book not available.\n");
}
```

```
void generateBill() {
    char title[MAX_NAME];
    int qty;
    double total = 0;
    printf("Enter book title and quantity (type 'end' to finish): \n");
    while (1) {
        scanf("%s", title);
        if (strcmp(title, "end") == 0) break;
        scanf("%d", &qty);
        Book *book = searchBook(root, title);
        if (book && book->quantity >= qty) {
            double cost = qty * book->price;
            printf("Title: %s, Quantity: %d, Cost: %.2f\n", book->title, qty,
cost);
            total += cost;
            book->quantity -= qty;
        } else {
            printf("Book not available or insufficient stock.\n");
        }
    }
    printf("Total Amount: %.2f\n", total);
}

void loadBooks() {
    FILE *file = fopen("books.txt", "r");
    char title[MAX_NAME];
    double price;
    int quantity;
    while (fscanf(file, "%s %lf %d", title, &price, &quantity) != EOF) {
        addBook(title, price, quantity);
    }
    fclose(file);
}

int main() {
    int choice;
    char title[MAX_NAME];
    double price;
    int quantity;
    loadBooks();
    printf("Admin Login\n");
    while (1) {
        printf("\n1. Add Book\n2. Delete Book\n3. Update Quantity\n4. Check
Availability\n5. Generate Bill\n6. Exit\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter title, price, quantity: ");
                scanf("%s %lf %d", title, &price, &quantity);
                addBook(title, price, quantity);
                break;
            case 2:
```

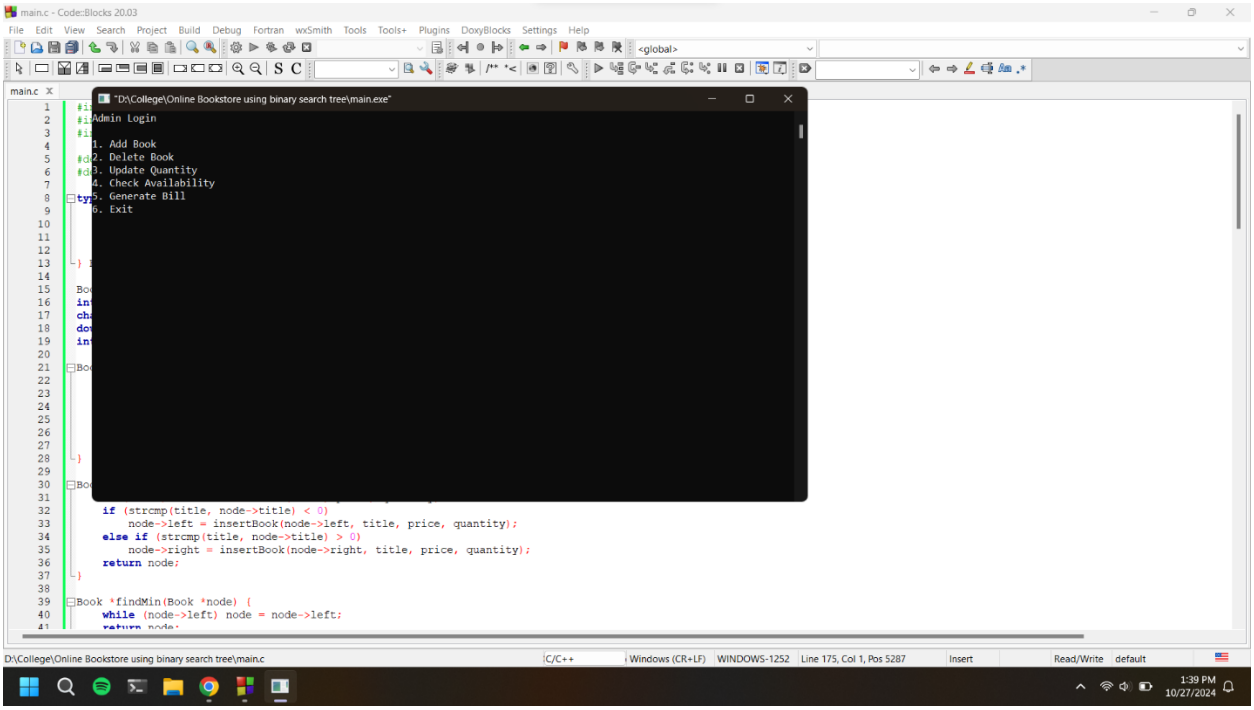
```
            printf("Enter title to delete: ");
            scanf("%s", title);
            root = deleteBook(root, title);
            break;
        case 3:
            printf("Enter title and new quantity: ");
            scanf("%s %d", title, &quantity);
            updateQuantity(title, quantity);
            break;
        case 4:
            printf("Enter title to check availability: ");
            scanf("%s", title);
            checkAvailability(title);
            break;
        case 5:
            generateBill();
            break;
        case 6:
            exit(0);
    }
}
return 0;
}
```

Output

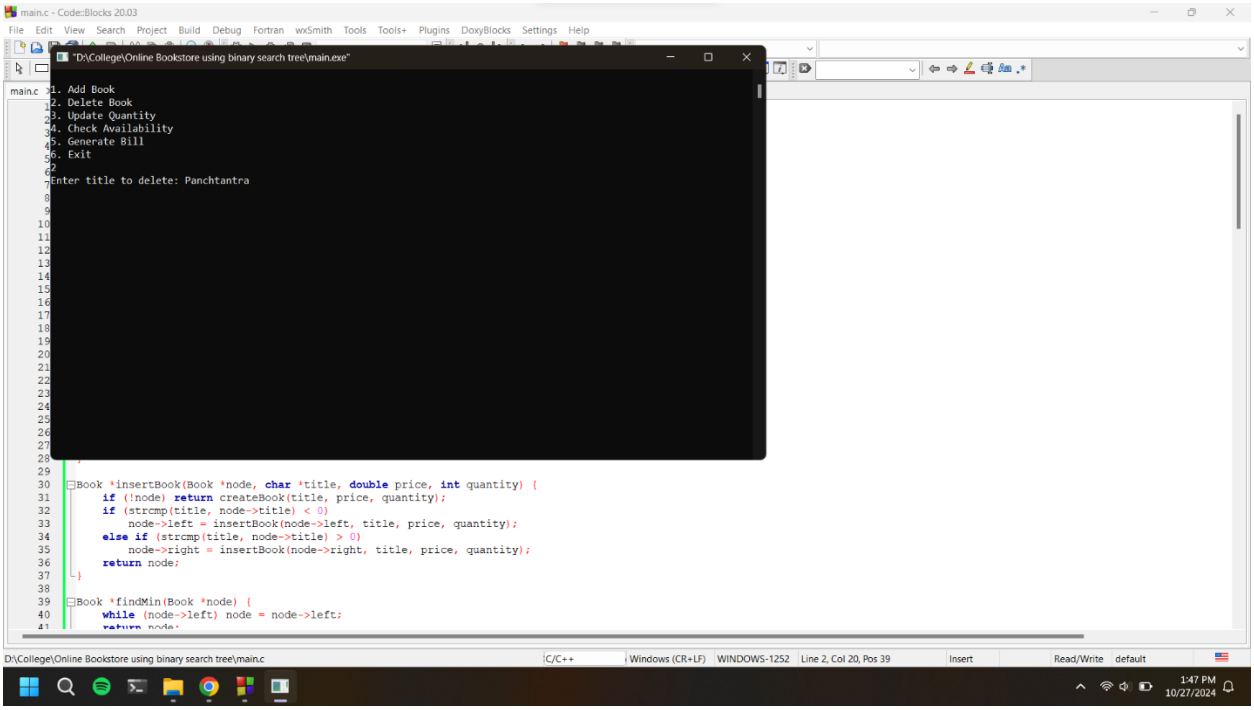
1. Admin Menu



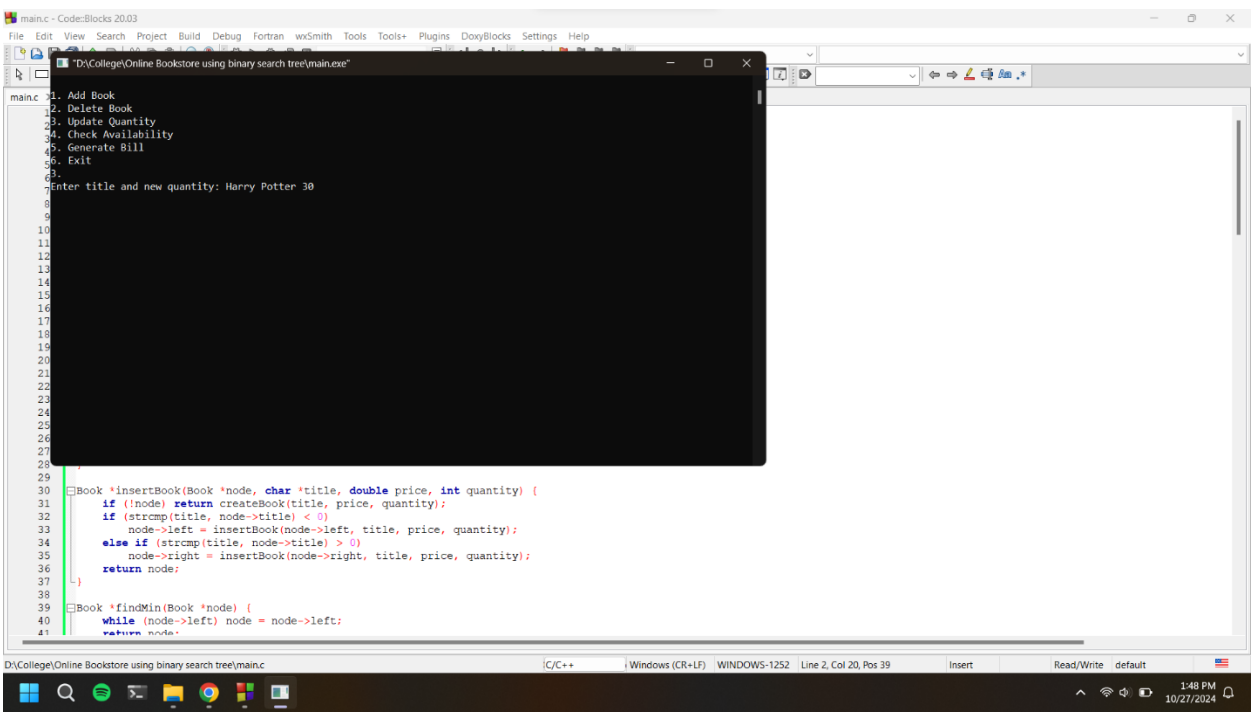
2. Add Book



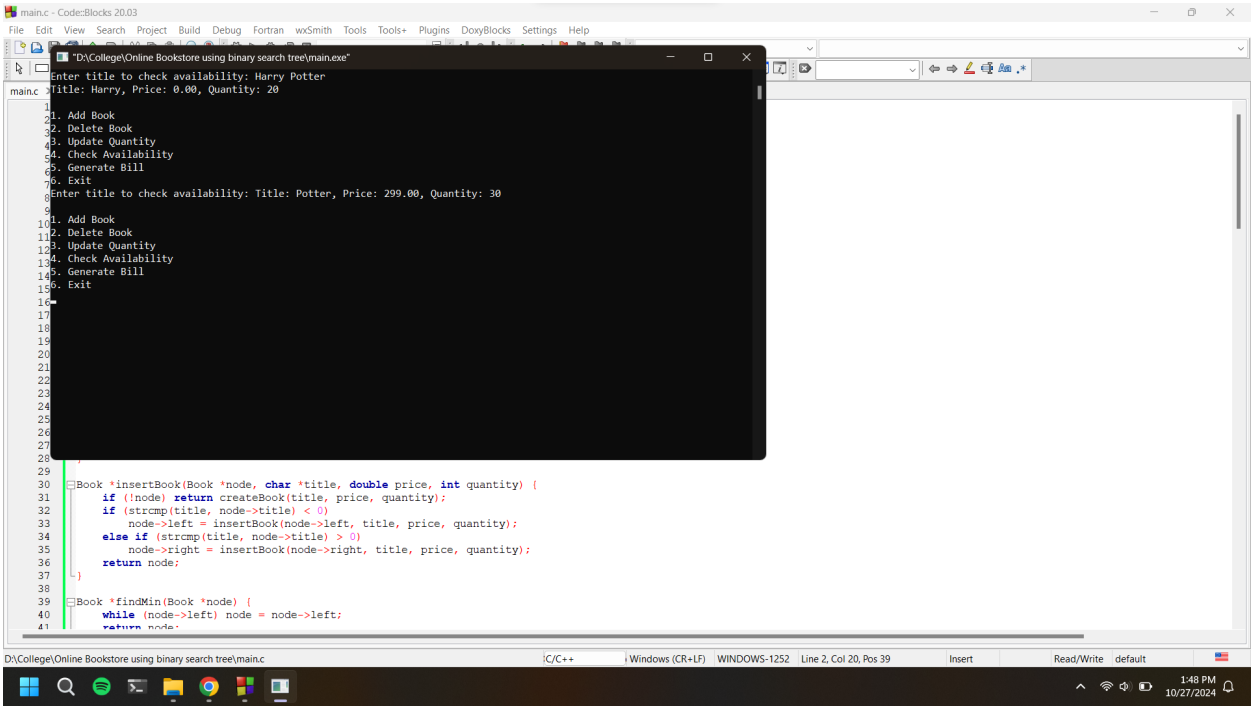
3. Delete Book



4. Update Quantity



5. Check availability



6. Generate Bill