```cpp
#include <iostream>
#include <cstring>
#include <iomanip>

using namespace std;

// Constants
const int MAX_STUDENTS = 20;
const int MAX_BOOKS = 15;
const int MAX_NAME_LENGTH = 50;
// Global variables
int student_count = 0;
int book_count = 0;
double student_balance[MAX_STUDENTS];
int student_roll[MAX_STUDENTS];
char student_name[MAX_STUDENTS][MAX_NAME_LENGTH];
char book_title[MAX_BOOKS][MAX_NAME_LENGTH];
char book_author[MAX_BOOKS][MAX_NAME_LENGTH];
int book_isbn[MAX_BOOKS];
bool book_available[MAX_BOOKS];

// Function prototypes
void create_account();
void display(int roll);
void deposit_amount(int roll, double amount);
void issue_item(int roll);
void display_sorted();
int find_student(int roll);
int find_book(int isbn);
void add_book();
void edit_book();
void view_books();

int main() {
    // Initialization
    // Add initial 15 books to the library
    // TODO: Replace with actual book data
    for (int i = 0; i < MAX_BOOKS; i++) {
        strcpy(book_title[i], "Title");
        strcpy(book_author[i], "Author");
        book_isbn[i] = i + 1000;
        book_available[i] = true;
    }
    book_count = MAX_BOOKS;

    int option;
    bool is_admin;
    string password;

    while (true) {
        cout << "Login as:\n1. Admin\n2. Student\n0. Exit\n";
        cin >> option;

        if (option == 0) {
            break;
```

```
56:         }
57:
58:         is_admin = (option == 1);
59:
60:         cout << "Enter password: ";
61:         cin >> password;
62:
63:         if (password == "password") { // Use a simple password for demonstration purposes.
64:             if (is_admin) {
65:                 cout << "Admin options:\n1. Add book\n2. Edit book\n3. View book status\n4. V
66:                 cin >> option;
67:
68:                 switch (option) {
69:                     case 1: {
70:                         add_book();
71:                         break;
72:                     }
73:                     case 2: {
74:                         edit_book();
75:                         break;
76:                     }
77:                     case 3: {
78:                         view_books();
79:                         break;
80:                     }
81:                     case 4: {
82:                         display_sorted();
83:                         break;
84:                     }
85:                     case 5: {
86:                         int roll;
87:                         cout << "Enter student roll number: ";
88:                         cin >> roll;
89:                         display(roll);
90:                         break;
91:                     }
92:                 }
93:             } else {
94:                 int roll;
95:                 cout << "Enter your roll number: ";
96:                 cin >> roll;
97:
98:                 int index = find_student(roll);
99:                 if (index == -1) {
100:                    cout << "Student not found. Create an account? (1. Yes / 2. No): ";
101:                    cin >> option;
102:                    if (option == 1) {
103:                        create_account();
104:                    }
105:                } else {
106:                    cout << "Student options:\n1. View balance\n2. Deposit amount\n3. Issue i
107:                    cin >> option;
108:
109:                    switch (option) {
110:                        case 1: {
```

```
                            display(roll);
                            break;
                        }
                        case 2: {
                            double amount;
                            cout << "Enter the amount to deposit: ";
                            cin >> amount;
                            deposit_amount(roll, amount);
                            break;
                        }
                        case 3: {
                            issue_item(roll);
                            break;
                        }
                    }
                }
            }
        } else {
            cout << "Incorrect password.\n";
}
}
return 0;
}

void create_account() {
if (student_count >= MAX_STUDENTS) {
cout << "Student limit reached. Cannot create more accounts.\n";
return;
}

int roll;
cout << "Enter roll number (BBRRRR format): ";
cin >> roll;

if (find_student(roll) != -1) {
    cout << "Account already exists for this roll number.\n";
    return;
}

student_roll[student_count] = roll;
cout << "Enter student name: ";
cin.ignore();
cin.getline(student_name[student_count], MAX_NAME_LENGTH);

double initial_deposit;
cout << "Enter initial deposit amount ($50 minimum): ";
cin >> initial_deposit;

if (initial_deposit < 50) {
    cout << "Initial deposit must be at least $50.\n";
    return;
}

student_balance[student_count] = initial_deposit - 20 - 30; // Account opening and security d
student_count++;
```

```
166: }
167:
168: void display(int roll) {
169: int index = find_student(roll);
170: if (index == -1) {
171:     cout << "Student not found.\n";
172:     return;
173: }
174:
175: cout << "Roll No: " << student_roll[index] << endl;
176: cout << "Name: " << student_name[index] << endl;
177: cout << "Balance: $" << fixed << setprecision(2) << student_balance[index] << endl;
178: }
179:
180: void deposit_amount(int roll, double amount) {
181: int index = find_student(roll);
182: if (index == -1) {
183:     cout << "Student not found.\n";
184:     return;
185: }
186:
187: student_balance[index] += amount;
188: cout << "New balance: $" << fixed << setprecision(2) << student_balance[index] << endl;
189: }
190:
191: void issue_item(int roll) {
192: int index = find_student(roll);
193: if (index == -1) {
194:     cout << "Student not found.\n";
195:     return;
196: }
197:
198: cout << "Available books:\n";
199: for (int i = 0; i < book_count; i++) {
200:     if (book_available[i]) {
201:         cout << i + 1 << ". " << book_title[i] << " by " << book_author[i] << " (ISBN: " << b
202:     }
203: }
204:
205: int choice;
206: cout << "Enter the number of the book you want to issue (0 to cancel): ";
207: cin >> choice;
208:
209: if (choice == 0) {
210:     return;
211: }
212:
213: if (book_available[choice - 1] && student_balance[index] >= 2) {
214:     book_available[choice - 1] = false;
215:     student_balance[index] -= 2;
216:     cout << "Book issued successfully. New balance: $" << fixed << setprecision(2) << student
217: } else {
218:     cout << "Cannot issue the book. Insufficient balance or book is unavailable.\n";
219: }
220: }
```

```
221:
222: void display_sorted() {
223: for (int i = 0; i < student_count; i++) {
224: for (int j = i + 1; j < student_count; j++) {
225: if (student_roll[i] > student_roll[j]) {
226: swap(student_roll[i], student_roll[j]);
227: swap(student_balance[i], student_balance[j]);
228: swap(student_name[i], student_name[j]);
229: }
230: }
231: }
232:
233: for (int i = 0; i < student_count; i++) {
234:     cout << student_roll[i]<< " - " << student_name[i] << " - Balance: $" << fixed << setprec
235: }
236: }
237:
238: int find_student(int roll) {
239: for (int i = 0; i < student_count; i++) {
240: if (student_roll[i] == roll) {
241: return i;
242: }
243: }
244: return -1;
245: }
246:
247: int find_book(int isbn) {
248: for (int i = 0; i < book_count; i++) {
249: if (book_isbn[i] == isbn) {
250: return i;
251: }
252: }
253: return -1;
254: }
255:
256: void add_book() {
257: if (book_count >= MAX_BOOKS) {
258: cout << "Book limit reached. Cannot add more books.\n";
259: return;
260: }
261: cout << "Enter book title: ";
262: cin.ignore();
263: cin.getline(book_title[book_count], MAX_NAME_LENGTH);
264:
265: cout << "Enter book author: ";
266: cin.getline(book_author[book_count], MAX_NAME_LENGTH);
267:
268: int isbn;
269: cout << "Enter book ISBN: ";
270: cin >> isbn;
271:
272: if (find_book(isbn) != -1) {
273:     cout << "A book with this ISBN already exists.\n";
274:     return;
275: }
```

```
276:
277: book_isbn[book_count] = isbn;
278: book_available[book_count] = true;
279: book_count++;
280: }
281:
282: void edit_book() {
283: int isbn;
284: cout << "Enter book ISBN to edit: ";
285: cin >> isbn;
286: int index = find_book(isbn);
287: if (index == -1) {
288:     cout << "Book not found.\n";
289:     return;
290: }
291:
292: cout << "Current book title: " << book_title[index] << endl;
293: cout << "Enter new book title: ";
294: cin.ignore();
295: cin.getline(book_title[index], MAX_NAME_LENGTH);
296:
297: cout << "Current book author: " << book_author[index] << endl;
298: cout << "Enter new book author: ";
299: cin.getline(book_author[index], MAX_NAME_LENGTH);
300:
301: cout << "Book details updated.\n";
302: }
303:
304: void view_books() {
305: for (int i = 0; i < book_count; i++) {
306: cout << "Title: " << book_title[i] << endl;
307: cout << "Author: " << book_author[i] << endl;
308: cout << "ISBN: " << book_isbn[i] << endl;
309: cout << "Available: " << (book_available[i] ? "Yes" : "No") << endl << endl;
310: }
311: }
```