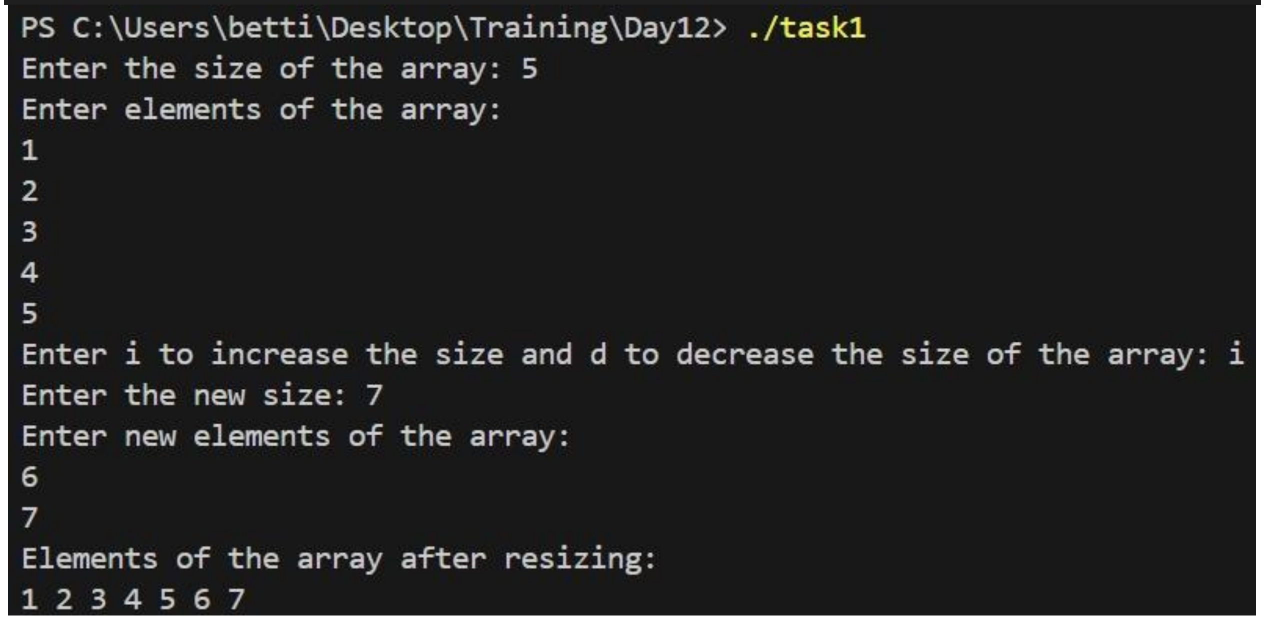
1.

```
/*Problem 1: Dynamic Array Resizing
Objective: Write a program to dynamically allocate an integer array and allow the
user to resize it.
Description:
The program should ask the user to enter the initial size of the array.
Allocate memory using malloc.
Allow the user to enter elements into the array.
Provide an option to increase or decrease the size of the array. Use realloc to
adjust the size.
Print the elements of the array after each resizing operation.*/



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

int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int *arr = (int*)malloc(size * sizeof(int));
    if (arr == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    printf("Enter elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    char op;
    printf("Enter i to increase the size and d to decrease the size of the array:
");
    scanf(" %c", &op);
    int newSize;

    switch(op) {
        case 'i':
            printf("Enter the new size: ");
            scanf("%d", &newSize);
            if(newSize <= size) {
                printf("New size is equal to or less than the current size\n");
                return 1;
            }
            arr = realloc(arr, newSize * sizeof(int));
            if (arr == NULL) {
                printf("Memory reallocation failed\n");
                return 1;
            }
            printf("Enter new elements of the array:\n");
            for (int i = size; i < newSize; i++) {
                scanf("%d", &arr[i]);
            }
            size = newSize;
            break;
        case 'd':
            printf("Enter the new size: ");
            scanf("%d", &newSize);
            if(newSize >= size) {
                printf("New size is equal to or greater than the current
size\n");
                return 1;
            }
            arr = realloc(arr, newSize * sizeof(int));
            if (arr == NULL) {
                printf("Memory reallocation failed\n");
                return 1;
            }
            size = newSize;
            break;
        default:
            printf("Invalid option\n");
            return 1;
    }

    printf("Elements of the array after resizing:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
```

```
}
```

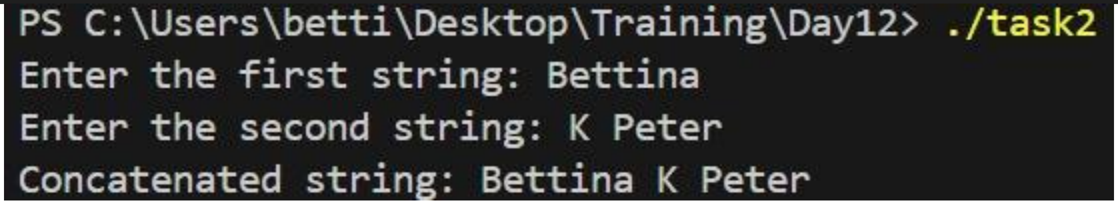


```
PS C:\Users\betti\Desktop\Training\Day12> ./task1
Enter the size of the array: 5
Enter elements of the array:
1
2
3
4
5
Enter i to increase the size and d to decrease the size of the array: i
Enter the new size: 7
Enter new elements of the array:
6
7
Elements of the array after resizing:
1 2 3 4 5 6 7
```

2.

```
/*Problem 2: String Concatenation Using Dynamic Memory
Objective: Create a program that concatenates two strings using dynamic memory
allocation.
Description:
Accept two strings from the user.
Use malloc to allocate memory for the first string.
Use realloc to resize the memory to accommodate the concatenated string.
Concatenate the strings and print the result.
Free the allocated memory.*/

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

int main() {
    char *str1 = (char*)malloc(100 * sizeof(char));
    if (str1 == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    char str2[100];

    printf("Enter the first string: ");
```

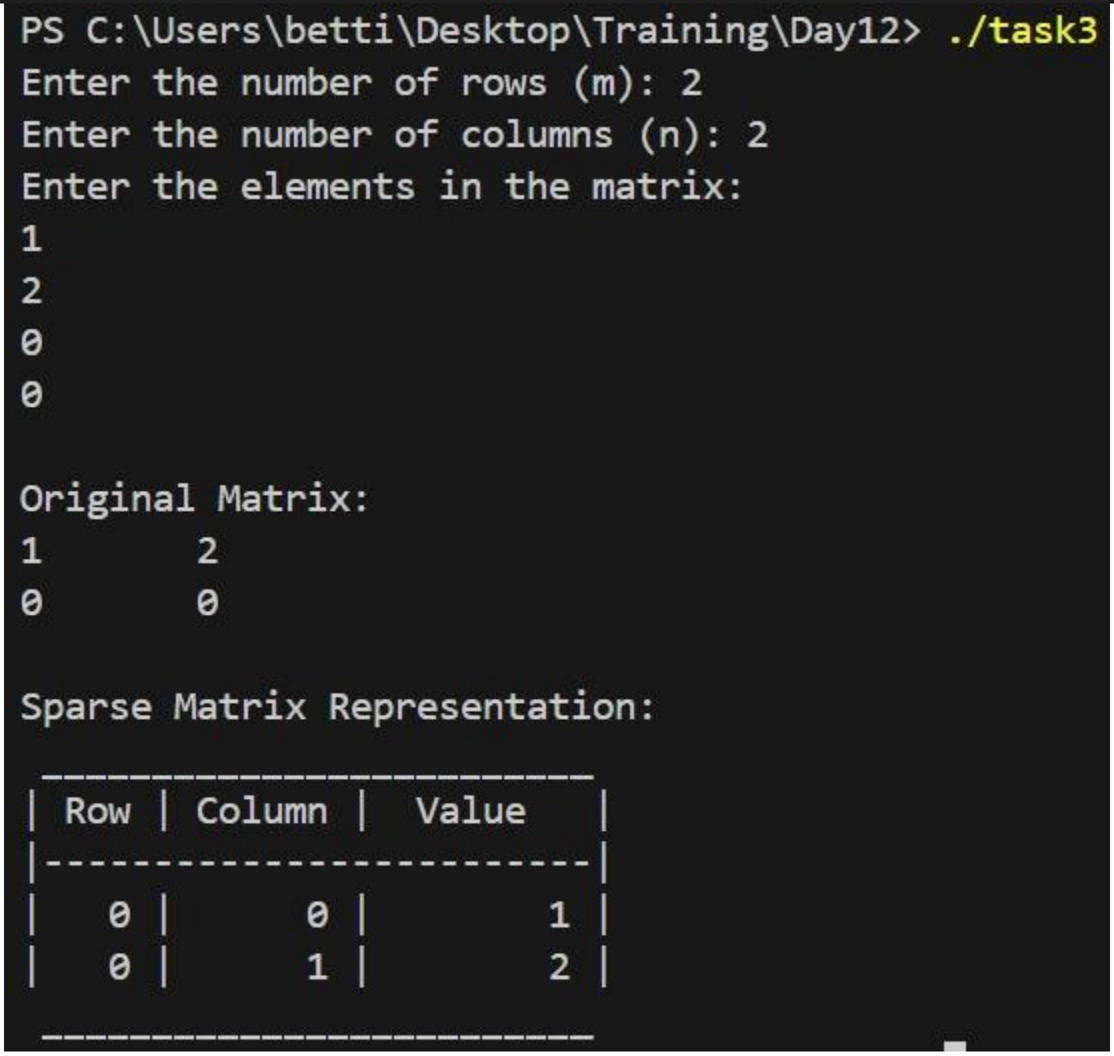
```c
    scanf("%[^\n]", str1);
    getchar();

    printf("Enter the second string: ");
    scanf("%[^\n]", str2);

    str1 = (char*)realloc(str1,(strlen(str2)+1));

    strcat(str1, str2);
    printf("Concatenated string: %s\n", str1);

    free(str1);
    return 0;
}
```



```
PS C:\Users\betti\Desktop\Training\Day12> ./task2
Enter the first string: Bettina
Enter the second string: K Peter
Concatenated string: Bettina K Peter
```

3.

```c
/*Problem 3: Sparse Matrix Representation
Objective: Represent a sparse matrix using dynamic memory allocation.
Description:
Accept a matrix of size m×nm \times nm×n from the user.
Store only the non-zero elements in a dynamically allocated array of
structures (with fields for row, column, and value).
Print the sparse matrix representation.
Free the allocated memory at the end.*/

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

int main() {
    int m, n;
    printf("Enter the number of rows (m): ");
    scanf("%d", &m);
    printf("Enter the number of columns (n): ");
    scanf("%d", &n);

    int matrix[m][n];
    int non_zero = 0;

    printf("Enter the elements in the matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
            if (matrix[i][j] != 0) {
                non_zero++;
            }
        }
    }

    printf("\nOriginal Matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    int *rows = malloc(non_zero * sizeof(int));
    int *cols = malloc(non_zero * sizeof(int));
    int *values = malloc(non_zero * sizeof(int));

    int count = 0;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j]!= 0) {
                rows[count] = i;
                cols[count] = j;
                values[count] = matrix[i][j];
                count++;
            }
        }
    }

    printf("\nSparse Matrix Representation:\n");
    printf(" _________________________\n");
    printf("| Row | Column |  Value   |\n");
    printf("|-------------------------|\n");
    for (int i = 0; i < non_zero; i++) {
        printf("| %3d | %6d | %8d |\n", rows[i], cols[i], values[i]);
    }
    printf(" _________________________\n");



    free(rows);
    free(cols);
    free(values);

    return 0;
}
```

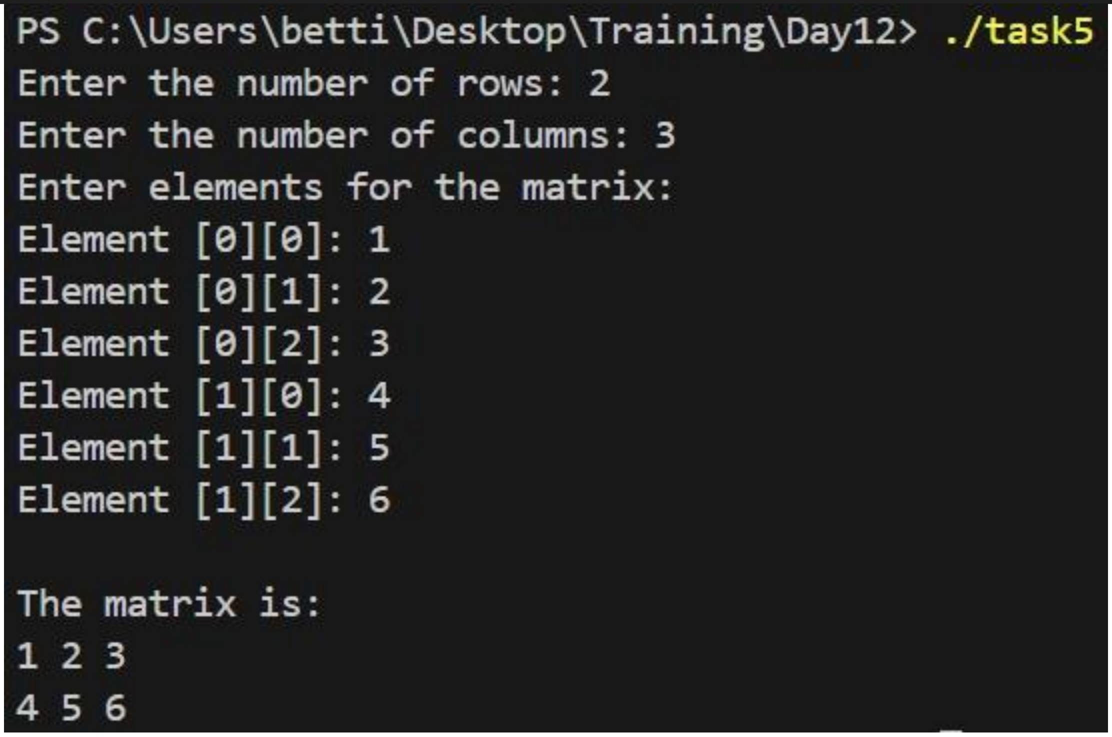


```
PS C:\Users\betti\Desktop\Training\Day12> ./task3
Enter the number of rows (m): 2
Enter the number of columns (n): 2
Enter the elements in the matrix:
1
2
0
0

Original Matrix:
1       2
0       0

Sparse Matrix Representation:

 _________________________
| Row | Column |  Value    |
|-------------------------|
|   0 |      0 |         1 |
|   0 |      1 |         2 |

 _________________________
```

4.

```c
/*Problem 5: Dynamic 2D Array Allocation
Objective: Write a program to dynamically allocate a 2D array.
Description:
```

```c
Accept the number of rows and columns from the user.
Use malloc (or calloc) to allocate memory for the rows and columns dynamically.
Allow the user to input values into the 2D array.
Print the array in matrix format.
Free all allocated memory at the end.*/

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

int main() {
    int rows, cols;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int **matrix = (int **)malloc(rows * sizeof(int *));
    if (matrix == NULL) {
        printf("Memory allocation failed for rows.\n");
        return 1;
    }

    for (int i = 0; i < rows; i++) {
        matrix[i] = (int *)malloc(cols * sizeof(int));
        if (matrix[i] == NULL) {
            printf("Memory allocation failed for row %d.\n", i);
            return 1;
        }
    }

    printf("Enter elements for the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("Element [%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }

    printf("\nThe matrix is:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }

    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;

}
```



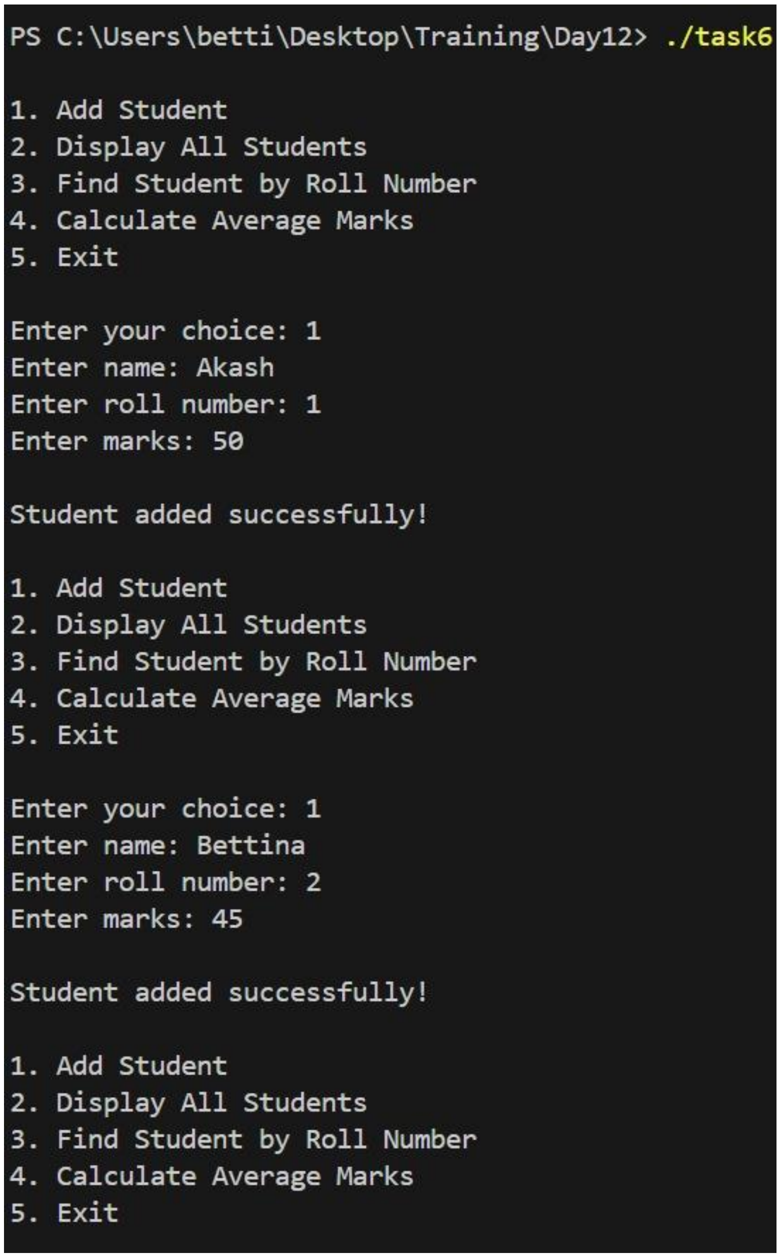
```
PS C:\Users\betti\Desktop\Training\Day12> ./task5
Enter the number of rows: 2
Enter the number of columns: 3
Enter elements for the matrix:
Element [0][0]: 1
Element [0][1]: 2
Element [0][2]: 3
Element [1][0]: 4
Element [1][1]: 5
Element [1][2]: 6

The matrix is:
1 2 3
4 5 6
```

5.

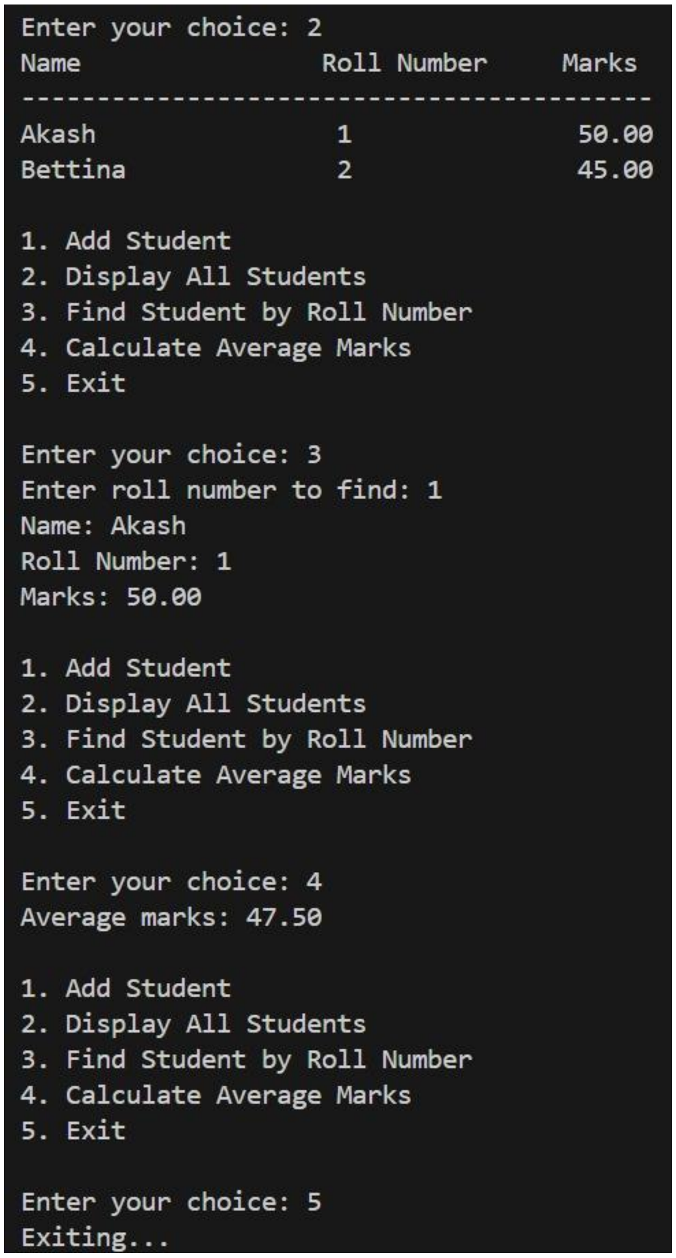
```c
//Student Record Management

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

struct student {
    char name[50];
    int rollNumber;
    float marks;
};

void AddStudent(struct student students[], int *numStudents);
void DisplayAll(struct student students[], int numStudents);
void FindStudent(struct student students[], int numStudents);
void AverageMarks(struct student students[], int numStudents);

int main() {
    struct student students[100];
    int choice;
    int numStudents = 0;

    while(1) {
        printf("\n1. Add Student\n2. Display All Students\n3. Find Student by Roll Number\n4. Calculate Average Marks\n5. Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                AddStudent(students, &numStudents);
                break;
            case 2:
                DisplayAll(students, numStudents);
                break;
            case 3:
                FindStudent(students, numStudents);
                break;
            case 4:
                AverageMarks(students, numStudents);
                break;
            case 5:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    }
    return 0;
}

void AddStudent(struct student students[], int *numStudents)
```

```c
{
    printf("Enter name: ");
    scanf("%s", &students[*numStudents].name);
    printf("Enter roll number: ");
    scanf("%d", &students[*numStudents].rollNumber);
    printf("Enter marks: ");
    scanf("%f", &students[*numStudents].marks);
    (*numStudents)++;
    printf("\nStudent added successfully!\n");
}

void DisplayAll(struct student students[], int numStudents)
{
    if(numStudents == 0) {
        printf("No students found.\n");
        return;
    }

    printf("Name                Roll Number     Marks\n");
    printf("------------------------------------------\n");

    for (int i = 0; i < numStudents; i++) {
        printf("%-20s %-15d %-10.2f\n", students[i].name, students[i].rollNumber,
students[i].marks);
    }
}

void FindStudent(struct student students[], int numStudents)
{
    int number;
    printf("Enter roll number to find: ");
    scanf("%d", &number);

    for(int i = 0; i < numStudents; i++) {
        if(students[i].rollNumber == number) {
            printf("Name: %s\nRoll Number: %d\nMarks: %0.2f\n", students[i].name,
students[i].rollNumber, students[i].marks);
            return;
        }
    }
    printf("Student not found.\n");
}
```

```c
void AverageMarks(struct student students[], int numStudents)
{
    if (numStudents == 0) {
        printf("No students available to calculate average marks.\n");
        return;
    }

    float sum = 0;
    for(int i = 0; i < numStudents; i++) {
        sum += students[i].marks;
    }
    printf("Average marks: %0.2f\n", sum/numStudents);
}
```



```
PS C:\Users\betti\Desktop\Training\Day12> ./task6

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 1
Enter name: Akash
Enter roll number: 1
Enter marks: 50

Student added successfully!

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 1
Enter name: Bettina
Enter roll number: 2
Enter marks: 45

Student added successfully!

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit
```



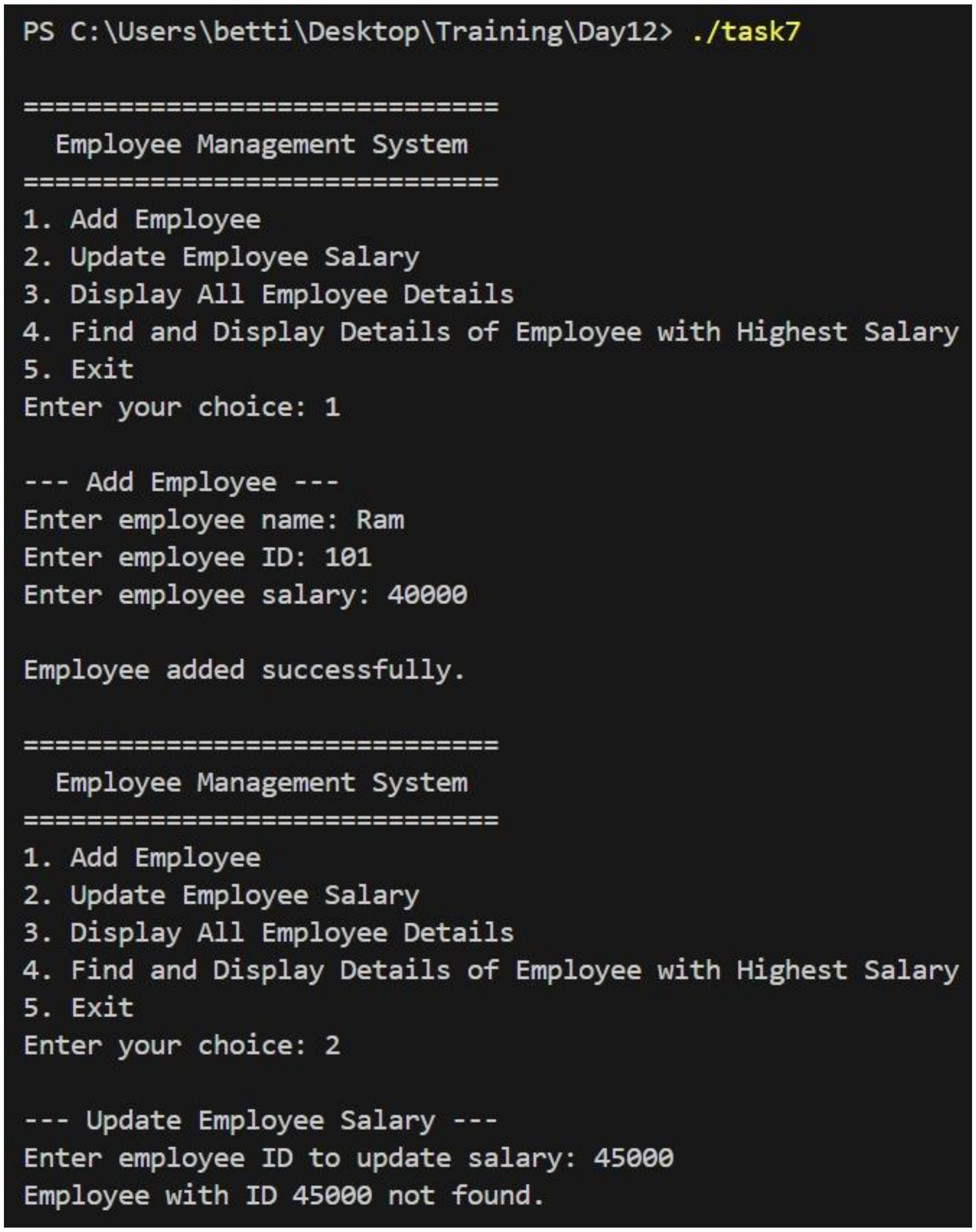
```
Enter your choice: 2
Name                    Roll Number     Marks
------------------------------------------
Akash                     1               50.00
Bettina                   2               45.00

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 3
Enter roll number to find: 1
Name: Akash
Roll Number: 1
Marks: 50.00

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 4
Average marks: 47.50

1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 5
Exiting...
```

6.

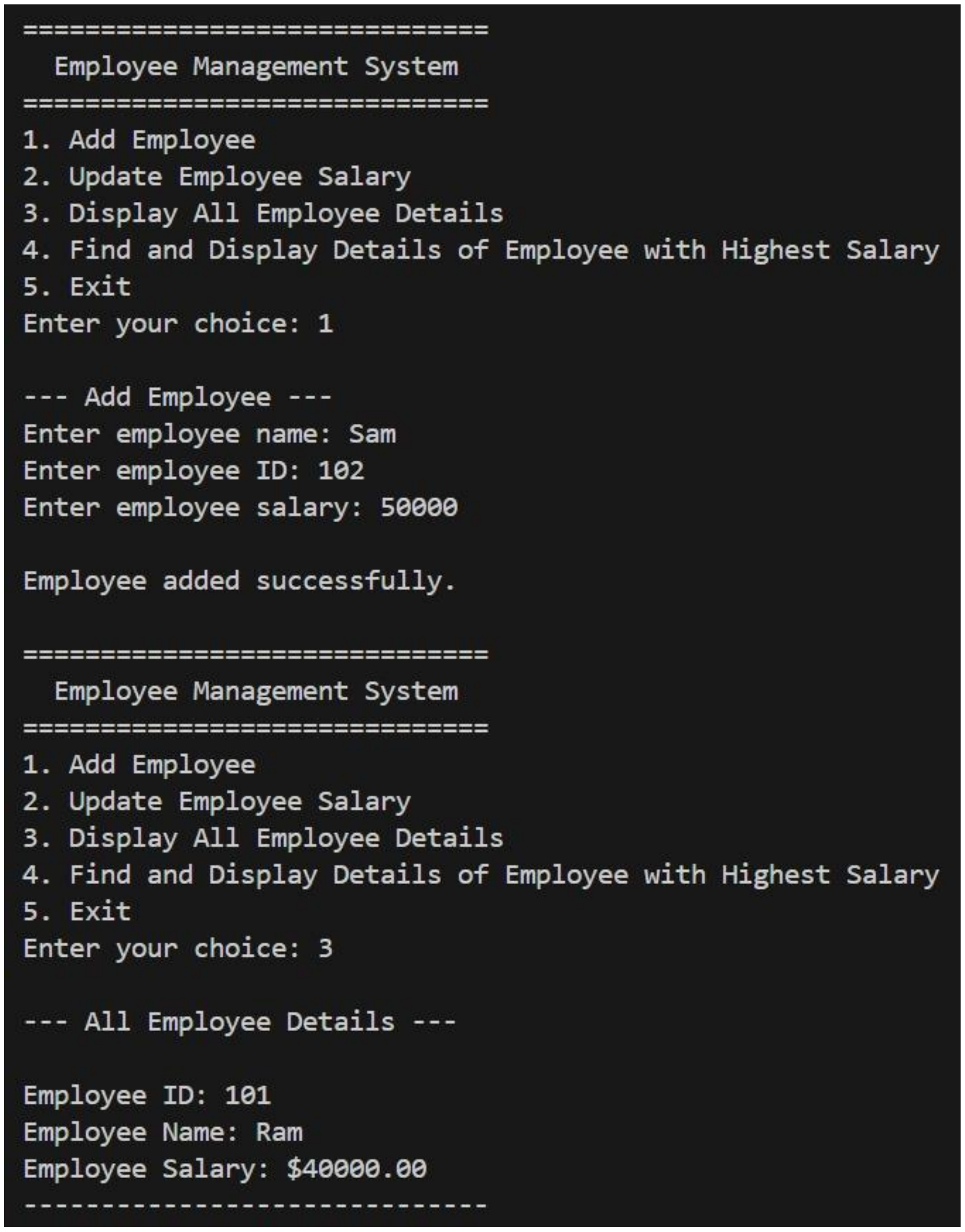
```
/*Problem 1: Employee Management System
Objective: Create a program to manage employee details using structures.
Description:
Define a structure Employee with fields:
int emp_id: Employee ID
char name[50]: Employee name
float salary: Employee salary
Write a menu-driven program to:
```

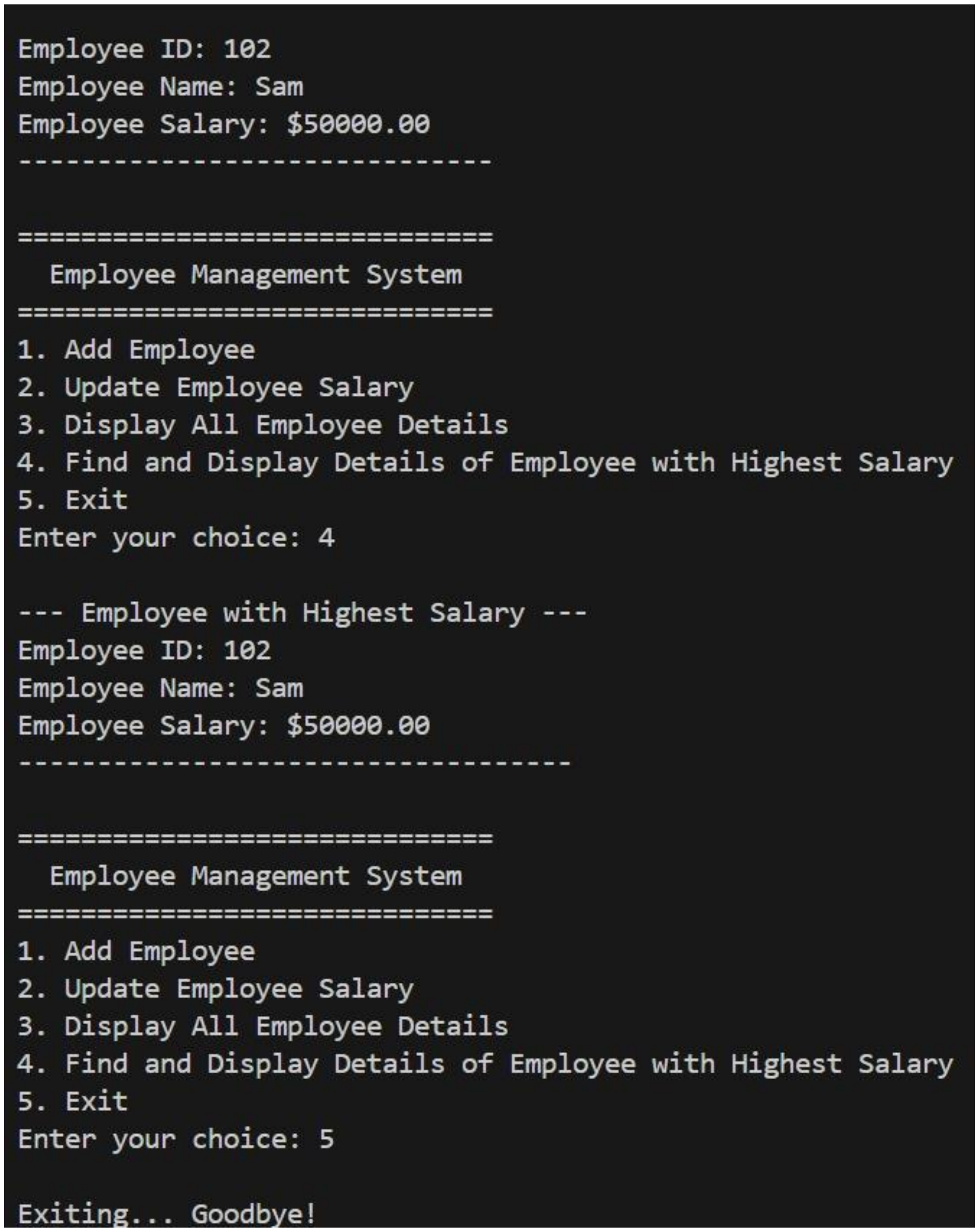
```c
Add an employee.
Update employee salary by ID.
Display all employee details.
Find and display details of the employee with the highest salary.*/

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

struct Employee {
    int emp_id;
    char name[50];
    float salary;
};

void addEmployee(struct Employee employees[], int* employeeCount);
void updateSalary(struct Employee employees[], int employeeCount);
void displayAll(struct Employee employees[], int employeeCount);
void highestSalary(struct Employee employees[], int employeeCount);

int main() {
    struct Employee employees[100];
    int employeeCount = 0;
    int choice;

    while (1) {
        printf("\n==============================\n");
        printf("  Employee Management System  \n");
        printf("==============================\n");
        printf("1. Add Employee\n");
        printf("2. Update Employee Salary\n");
        printf("3. Display All Employee Details\n");
        printf("4. Find and Display Details of Employee with Highest Salary\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addEmployee(employees, &employeeCount);
                break;
            case 2:
                updateSalary(employees, employeeCount);
                break;
            case 3:
                displayAll(employees, employeeCount);
                break;
            case 4:
                highestSalary(employees, employeeCount);
                break;
            case 5:
                printf("\nExiting... Goodbye!\n");
                return 0;
            default:
                printf("\nInvalid choice. Please try again.\n");
        }
    }
    return 0;
}

void addEmployee(struct Employee employees[], int* employeeCount) {
    printf("\n--- Add Employee ---\n");
    printf("Enter employee name: ");
    scanf("%s", employees[*employeeCount].name);
    printf("Enter employee ID: ");
    scanf("%d", &employees[*employeeCount].emp_id);
    printf("Enter employee salary: ");
    scanf("%f", &employees[*employeeCount].salary);
    (*employeeCount)++;
    printf("\nEmployee added successfully.\n");
}

void updateSalary(struct Employee employees[], int employeeCount) {
    int id;
    printf("\n--- Update Employee Salary ---\n");
    printf("Enter employee ID to update salary: ");
    scanf("%d", &id);
    for (int i = 0; i < employeeCount; i++) {
        if (employees[i].emp_id == id) {
            printf("Enter new salary: ");
            scanf("%f", &employees[i].salary);
            printf("Salary updated successfully.\n");
            return;
        }
    }
    printf("Employee with ID %d not found.\n", id);
}
```

```c
void displayAll(struct Employee employees[], int employeeCount) {
    if (employeeCount == 0) {
        printf("\nNo employees to display.\n");
        return;
    }
    printf("\n--- All Employee Details ---\n");
    for (int i = 0; i < employeeCount; i++) {
        printf("\nEmployee ID: %d\n", employees[i].emp_id);
        printf("Employee Name: %s\n", employees[i].name);
        printf("Employee Salary: $%.2f\n", employees[i].salary);
        printf("------------------------------\n");
    }
}

void highestSalary(struct Employee employees[], int employeeCount) {
    if (employeeCount == 0) {
        printf("\nNo employees available to find the highest salary.\n");
        return;
    }

    float highestSalary = employees[0].salary;
    int highestIndex = 0;
    for (int i = 1; i < employeeCount; i++) {
        if (employees[i].salary > highestSalary) {
            highestSalary = employees[i].salary;
            highestIndex = i;
        }
    }

    printf("\n--- Employee with Highest Salary ---\n");
    printf("Employee ID: %d\n", employees[highestIndex].emp_id);
    printf("Employee Name: %s\n", employees[highestIndex].name);
    printf("Employee Salary: $%.2f\n", employees[highestIndex].salary);
    printf("-----------------------------------\n");
}
```



```
PS C:\Users\betti\Desktop\Training\Day12> ./task7

==============================
  Employee Management System
==============================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 1

--- Add Employee ---
Enter employee name: Ram
Enter employee ID: 101
Enter employee salary: 40000

Employee added successfully.

==============================
  Employee Management System
==============================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 2

--- Update Employee Salary ---
Enter employee ID to update salary: 45000
Employee with ID 45000 not found.
```


```
================================
  Employee Management System
================================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 1

--- Add Employee ---
Enter employee name: Sam
Enter employee ID: 102
Enter employee salary: 50000

Employee added successfully.

================================
  Employee Management System
================================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 3

--- All Employee Details ---

Employee ID: 101
Employee Name: Ram
Employee Salary: $40000.00
------------------------------
```



```
Employee ID: 102
Employee Name: Sam
Employee Salary: $50000.00
------------------------------

==============================
  Employee Management System
==============================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 4

--- Employee with Highest Salary ---
Employee ID: 102
Employee Name: Sam
Employee Salary: $50000.00
-----------------------------------

==============================
  Employee Management System
==============================
1. Add Employee
2. Update Employee Salary
3. Display All Employee Details
4. Find and Display Details of Employee with Highest Salary
5. Exit
Enter your choice: 5

Exiting... Goodbye!
```

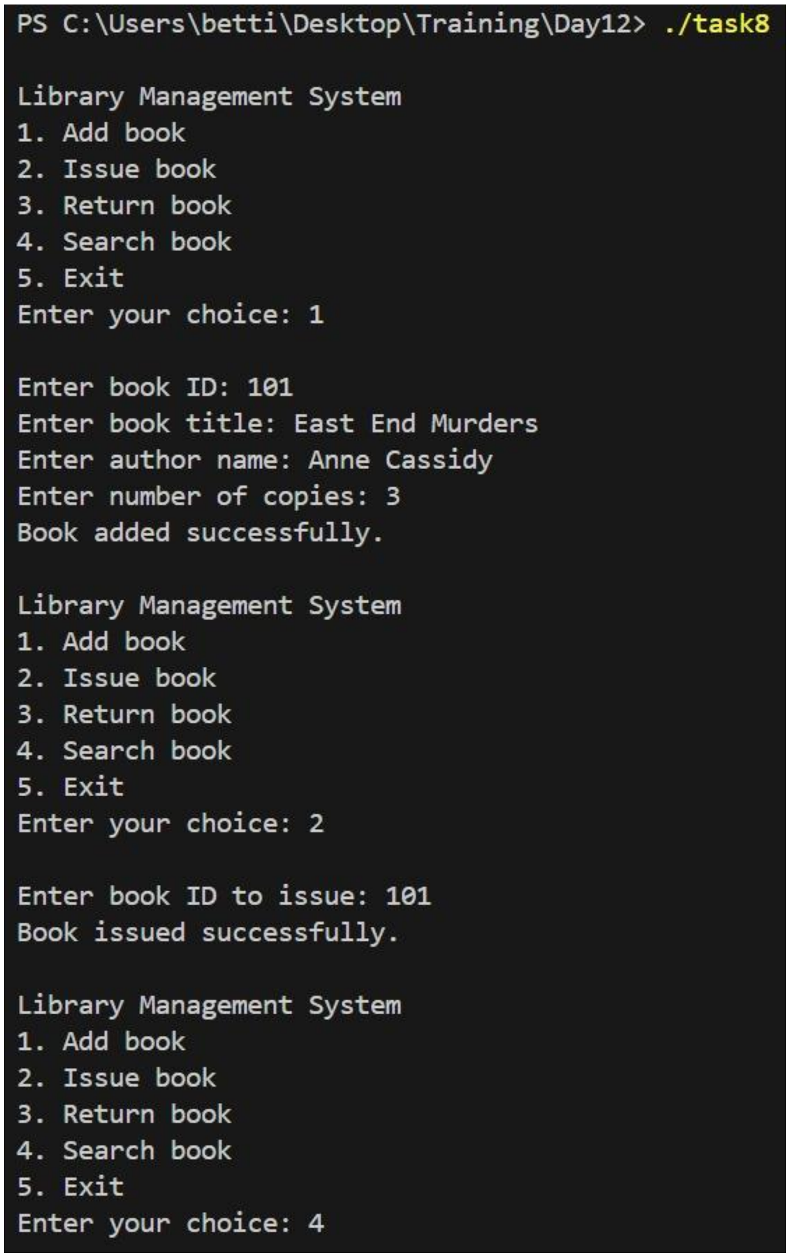
7.

```
/*Problem 2: Library Management System
Objective: Manage a library system with a structure to store book details.
Description:
Define a structure Book with fields:
int book_id: Book ID
char title[100]: Book title
char author[50]: Author name
int copies: Number of available copies
```

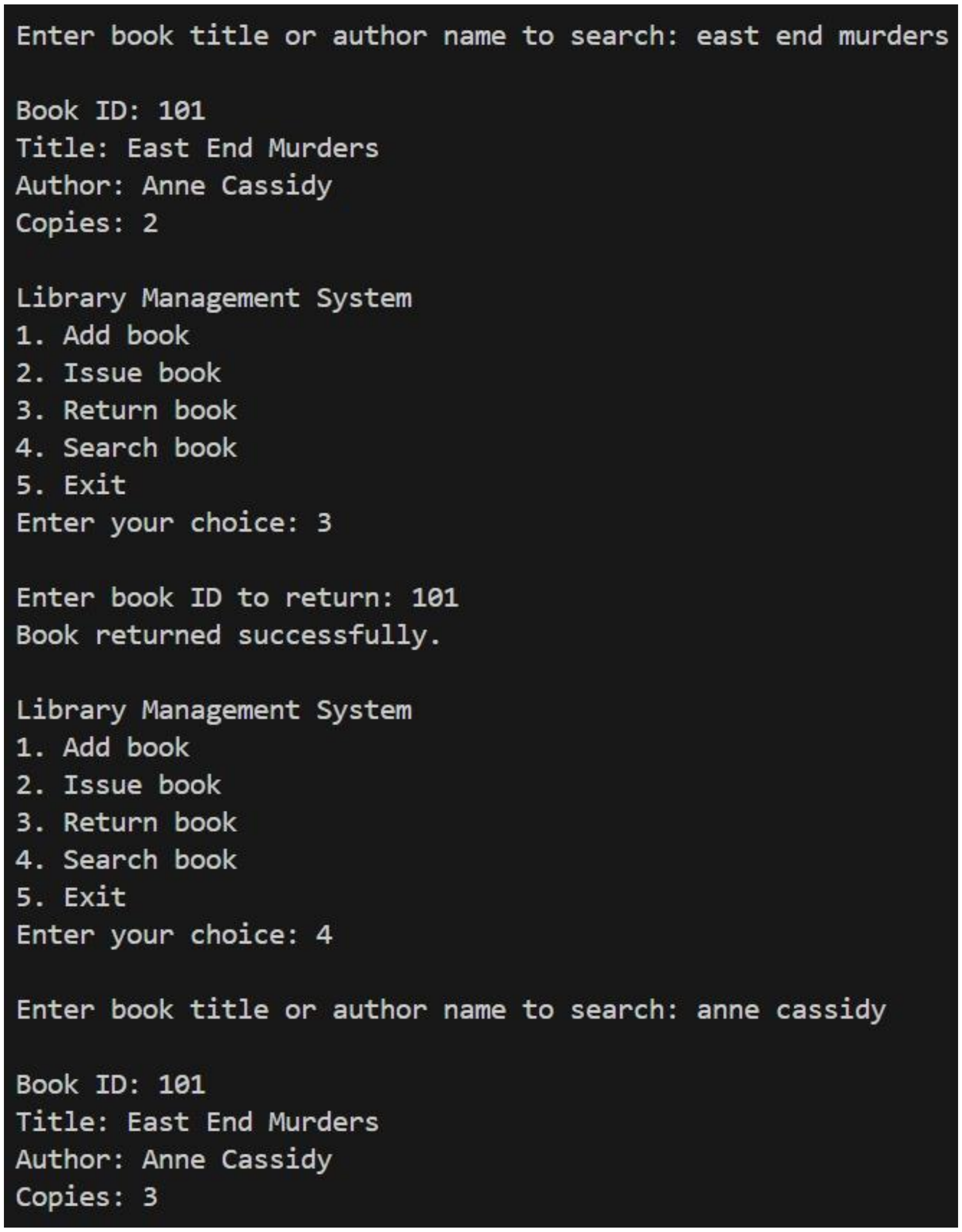
```c
Write a program to:
Add books to the library.
Issue a book by reducing the number of copies.
Return a book by increasing the number of copies.
Search for a book by title or author name.*/

#include <stdio.h>
#include <string.h>
#include <ctype.h>

struct Book {
    int book_id;
    char title[100];
    char author[50];
    int copies;
};

void addBook(struct Book books[], int *noOfBooks);
void issueBook(struct Book books[], int noOfBooks);
void returnBook(struct Book books[], int noOfBooks);
void searchBook(struct Book books[], int noOfBooks);

int main() {
    struct Book books[100];
    int noOfBooks = 0;
    int choice;

    while(1) {
        printf("\nLibrary Management System\n");
        printf("1. Add book\n");
        printf("2. Issue book\n");
        printf("3. Return book\n");
        printf("4. Search book\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        // Consume the newline character left by scanf
        getchar();

        switch(choice) {
            case 1:
                addBook(books, &noOfBooks);
                break;
            case 2:
                issueBook(books, noOfBooks);
                break;
            case 3:
                returnBook(books, noOfBooks);
                break;
            case 4:
                searchBook(books, noOfBooks);
                break;
            case 5:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void addBook(struct Book books[], int *noOfBooks) {
    printf("\nEnter book ID: ");
    scanf("%d", &books[*noOfBooks].book_id);

    // Consume the newline character left by scanf
    getchar();

    printf("Enter book title: ");
    fgets(books[*noOfBooks].title, sizeof(books[*noOfBooks].title), stdin);
    books[*noOfBooks].title[strcspn(books[*noOfBooks].title, "\n")] = '\0';  //
Remove the newline

    printf("Enter author name: ");
    fgets(books[*noOfBooks].author, sizeof(books[*noOfBooks].author), stdin);
    books[*noOfBooks].author[strcspn(books[*noOfBooks].author, "\n")] = '\0';  //
Remove the newline

    printf("Enter number of copies: ");
    scanf("%d", &books[*noOfBooks].copies);

    (*noOfBooks)++;
    printf("Book added successfully.\n");
```

```c
}

void issueBook(struct Book books[], int noOfBooks) {
    int bookId;
    printf("\nEnter book ID to issue: ");
    scanf("%d", &bookId);

    // Consume the newline character left by scanf
    getchar();

    for(int i = 0; i < noOfBooks; i++) {
        if(books[i].book_id == bookId) {
            if(books[i].copies > 0) {
                books[i].copies--;
                printf("Book issued successfully.\n");
                return;
            } else {
                printf("No copies available for this book.\n");
                return;
            }
        }
    }
    printf("Book not found.\n");
}

void returnBook(struct Book books[], int noOfBooks) {
    int bookId;
    printf("\nEnter book ID to return: ");
    scanf("%d", &bookId);

    // Consume the newline character left by scanf
    getchar();

    for(int i = 0; i < noOfBooks; i++) {
        if(books[i].book_id == bookId) {
            books[i].copies++;
            printf("Book returned successfully.\n");
            return;
        }
    }
    printf("Book not found.\n");
}
```

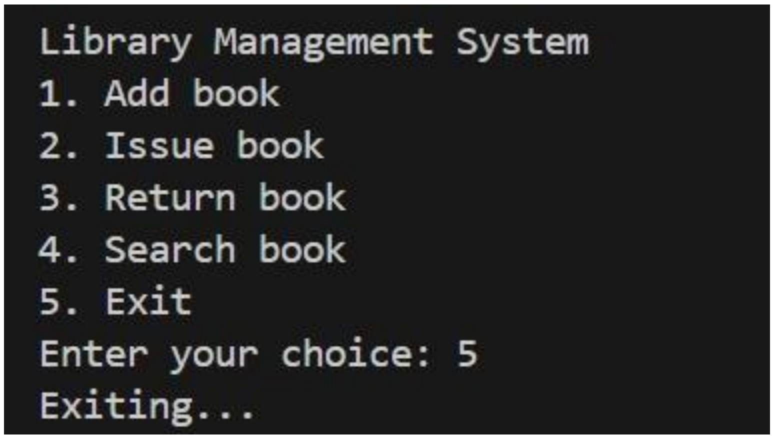
```
void searchBook(struct Book books[], int noOfBooks) {
    char query[100];
    printf("\nEnter book title or author name to search: ");
    fgets(query, sizeof(query), stdin);
    query[strcspn(query, "\n")] = '\0'; // Remove the newline

    int found = 0;
    for (int i = 0; i < noOfBooks; i++) {
        char titleLower[100], authorLower[50];
        for (int j = 0; books[i].title[j]; j++) {
            titleLower[j] = tolower(books[i].title[j]);
        }
        titleLower[strlen(books[i].title)] = '\0';

        for (int j = 0; books[i].author[j]; j++) {
            authorLower[j] = tolower(books[i].author[j]);
        }
        authorLower[strlen(books[i].author)] = '\0';

        if (strstr(titleLower, query) != NULL || strstr(authorLower, query) !=
NULL) {
            printf("\nBook ID: %d\n", books[i].book_id);
            printf("Title: %s\n", books[i].title);
            printf("Author: %s\n", books[i].author);
            printf("Copies: %d\n", books[i].copies);
            found = 1;
        }
    }

    if (!found) {
        printf("No books found matching the search query.\n");
    }
}
```



```
PS C:\Users\betti\Desktop\Training\Day12> ./task8

Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 1

Enter book ID: 101
Enter book title: East End Murders
Enter author name: Anne Cassidy
Enter number of copies: 3
Book added successfully.

Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 2

Enter book ID to issue: 101
Book issued successfully.

Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 4
```



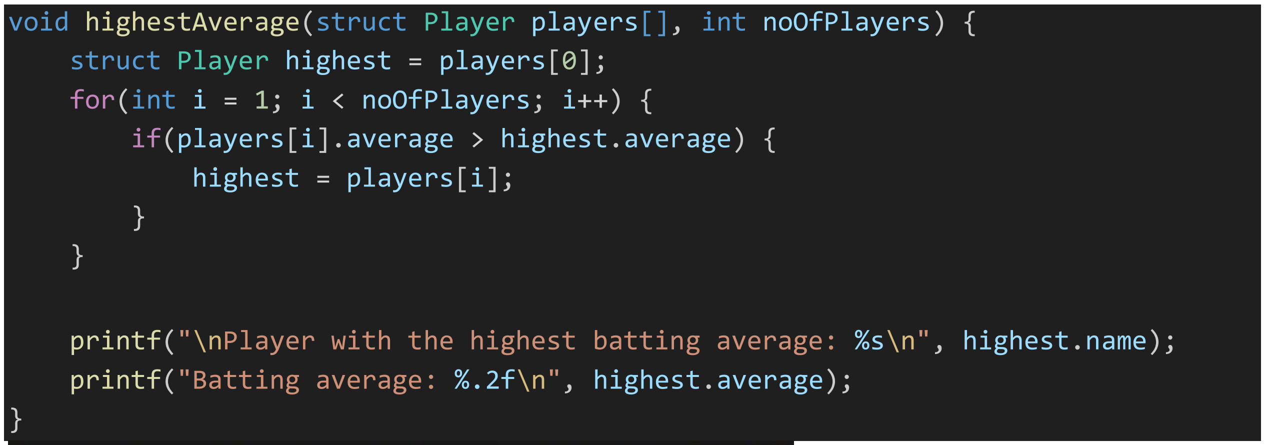
```
Enter book title or author name to search: east end murders

Book ID: 101
Title: East End Murders
Author: Anne Cassidy
Copies: 2

Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 3

Enter book ID to return: 101
Book returned successfully.

Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 4

Enter book title or author name to search: anne cassidy

Book ID: 101
Title: East End Murders
Author: Anne Cassidy
Copies: 3
```



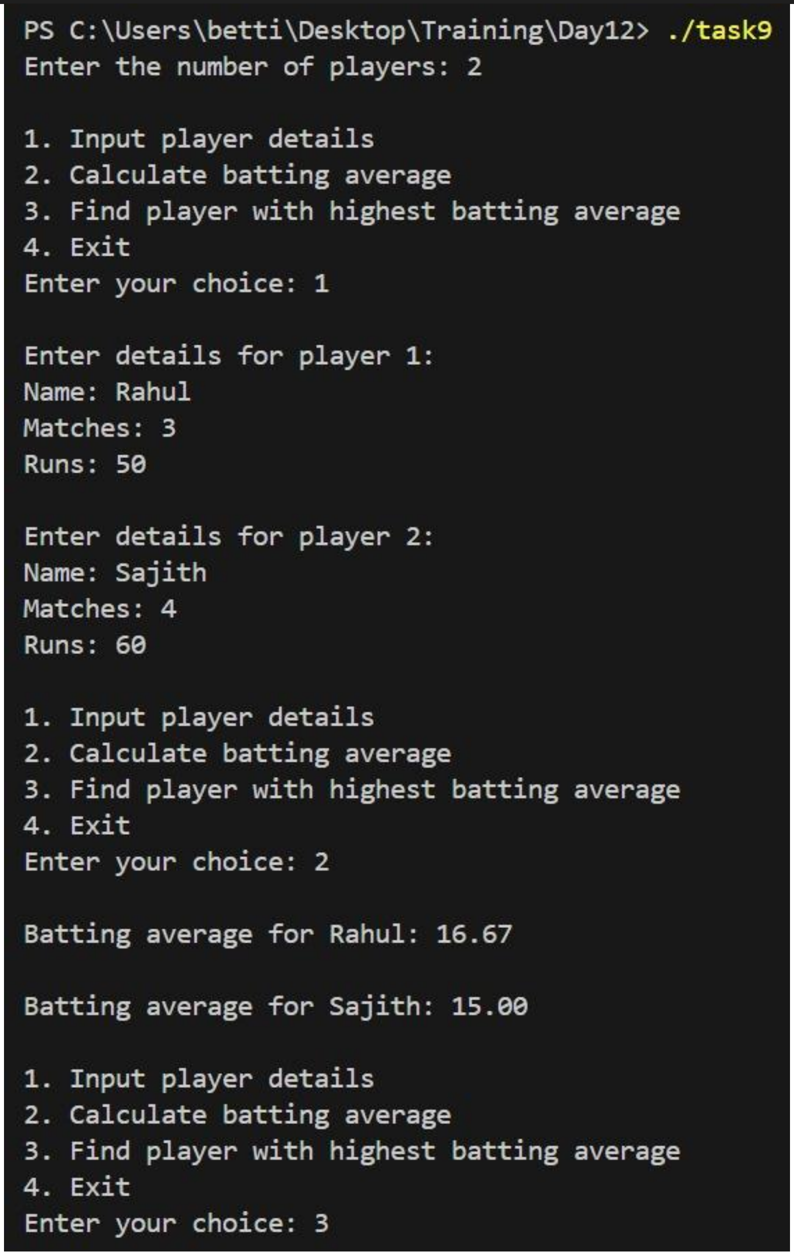
```
Library Management System
1. Add book
2. Issue book
3. Return book
4. Search book
5. Exit
Enter your choice: 5
Exiting...
```

8.

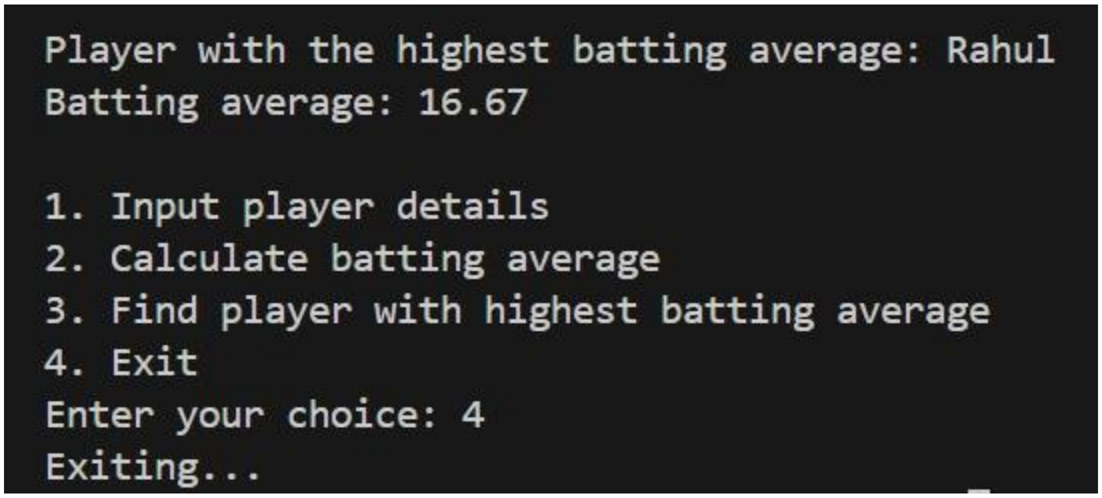
```c
/*Problem 3: Cricket Player Statistics
Objective: Store and analyze cricket player performance data.
Description:
Define a structure Player with fields:
char name[50]: Player name
int matches: Number of matches played
int runs: Total runs scored
float average: Batting average
Write a program to:
Input details for n players.
Calculate and display the batting average for each player.
Find and display the player with the highest batting average.*/

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

struct Player {
    char name[50];
    int matches;
    int runs;
    float average;
};

void addDetails(struct Player players[], int noOfPlayers);
void battingAverage(struct Player players[], int noOfPlayers);
void highestAverage(struct Player players[], int noOfPlayers);

int main() {
    int noOfPlayers;
    printf("Enter the number of players: ");
    scanf("%d", &noOfPlayers);

    struct Player players[noOfPlayers];

    int choice;
    while(1) {
        printf("\n1. Input player details\n2. Calculate batting average\n3. Find 
player with highest batting average\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                addDetails(players, noOfPlayers);
                break;
            case 2:
                battingAverage(players, noOfPlayers);
                break;
            case 3:
                highestAverage(players, noOfPlayers);
                break;
            case 4:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void addDetails(struct Player players[], int noOfPlayers) {
    for(int i = 0; i < noOfPlayers; i++) {
        printf("\nEnter details for player %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", players[i].name);
        printf("Matches: ");
        scanf("%d", &players[i].matches);
        printf("Runs: ");
        scanf("%d", &players[i].runs);
    }
}

void battingAverage(struct Player players[], int noOfPlayers) {
    for(int i = 0; i < noOfPlayers; i++) {
        players[i].average = (float)players[i].runs / players[i].matches;
        printf("\nBatting average for %s: %.2f\n", players[i].name,
players[i].average);
    }
}
```



```c
void highestAverage(struct Player players[], int noOfPlayers) {
    struct Player highest = players[0];
    for(int i = 1; i < noOfPlayers; i++) {
        if(players[i].average > highest.average) {
            highest = players[i];
        }
    }

    printf("\nPlayer with the highest batting average: %s\n", highest.name);
    printf("Batting average: %.2f\n", highest.average);
}
```



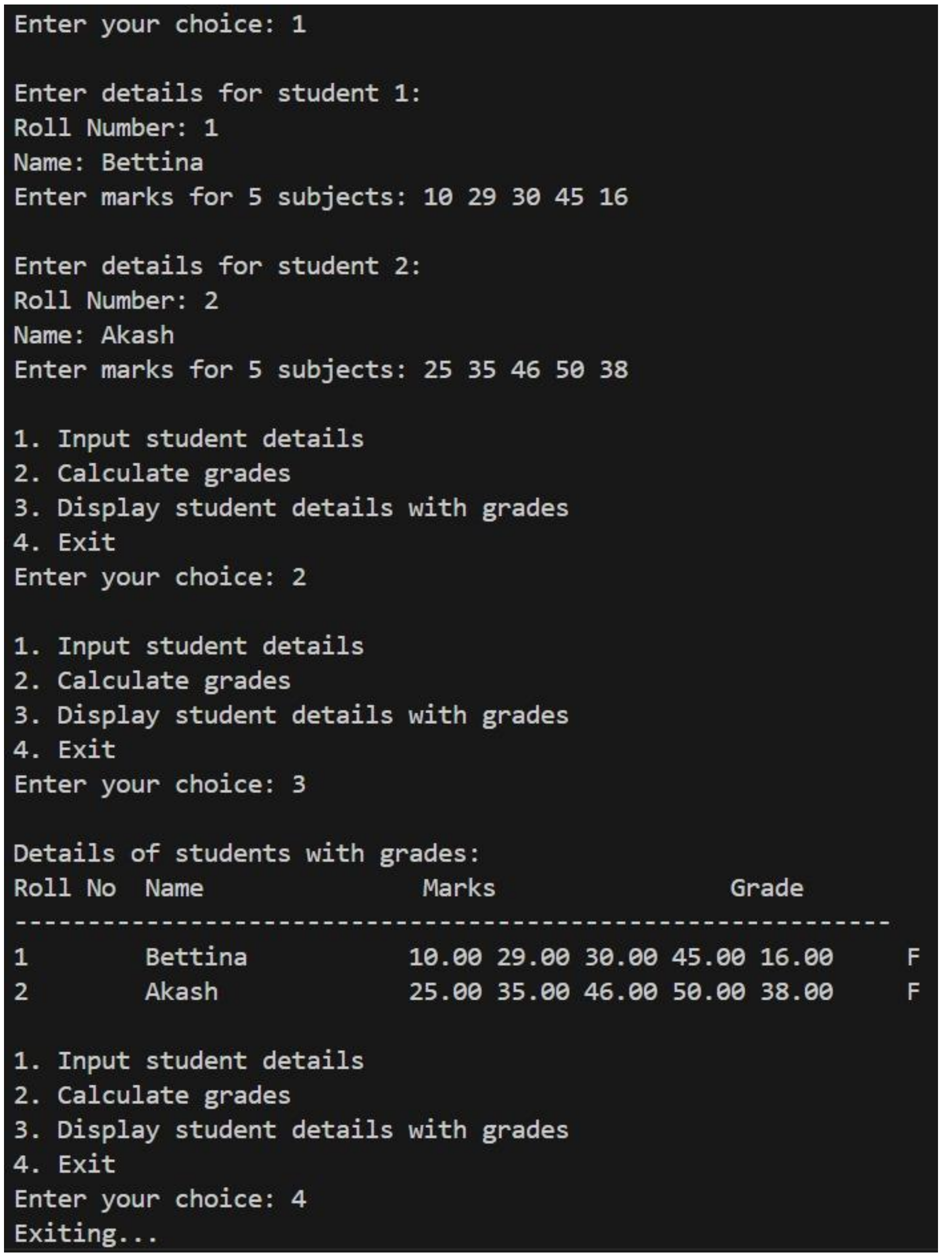
```
PS C:\Users\betti\Desktop\Training\Day12> ./task9
Enter the number of players: 2

1. Input player details
2. Calculate batting average
3. Find player with highest batting average
4. Exit
Enter your choice: 1

Enter details for player 1:
Name: Rahul
Matches: 3
Runs: 50

Enter details for player 2:
Name: Sajith
Matches: 4
Runs: 60

1. Input player details
2. Calculate batting average
3. Find player with highest batting average
4. Exit
Enter your choice: 2

Batting average for Rahul: 16.67

Batting average for Sajith: 15.00

1. Input player details
2. Calculate batting average
3. Find player with highest batting average
4. Exit
Enter your choice: 3
```



```
Player with the highest batting average: Rahul
Batting average: 16.67

1. Input player details
2. Calculate batting average
3. Find player with highest batting average
4. Exit
Enter your choice: 4
Exiting...
```

9.

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

struct Student {
    int roll_no;
    char name[50];
    float marks[5];
    char grade;
};

void addDetails(struct Student students[], int noOfStudents);
void calculateGrade(struct Student students[], int noOfStudents);
void displayDetails(struct Student students[], int noOfStudents);

int main() {
    int noOfStudents;
    printf("Enter the number of students: ");
    scanf("%d", &noOfStudents);

    struct Student students[noOfStudents];

    int choice;
    while(1) {
        printf("\n1. Input student details\n2. Calculate grades\n3. Display
student details with grades\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                addDetails(students, noOfStudents);
                break;
            case 2:
                calculateGrade(students, noOfStudents);
                break;
            case 3:
                displayDetails(students, noOfStudents);
                break;
            case 4:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void addDetails(struct Student students[], int noOfStudents) {
    for(int i = 0; i < noOfStudents; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Roll Number: ");
        scanf("%d", &students[i].roll_no);
        printf("Name: ");
        scanf("%s", students[i].name);

        printf("Enter marks for 5 subjects: ");
        for(int j = 0; j < 5; j++) {
            scanf("%f", &students[i].marks[j]);
        }
    }
}

void calculateGrade(struct Student students[], int noOfStudents) {
    for(int i = 0; i < noOfStudents; i++) {
        float sum = 0;
        for(int j = 0; j < 5; j++) {
            sum += students[i].marks[j];
        }
        float average = sum / 5;

        // Assign grade based on average marks
        if(average >= 90) {
            students[i].grade = 'A';
        } else if(average >= 80) {
            students[i].grade = 'B';
        } else if(average >= 70) {
            students[i].grade = 'C';
        } else if(average >= 60) {
            students[i].grade = 'D';
        } else {
            students[i].grade = 'F';
        }
    }
}

void displayDetails(struct Student students[], int noOfStudents) {
    printf("\nDetails of students with grades:\n");
    printf("Roll No  Name               Marks                Grade\n");
    printf("------------------------------------------------------------\n");

    for(int i = 0; i < noOfStudents; i++) {
        printf("%-8d %-18s", students[i].roll_no, students[i].name);
        for(int j = 0; j < 5; j++) {
            printf("%.2f ", students[i].marks[j]);
        }
        printf("    %c\n", students[i].grade);
    }
}
```

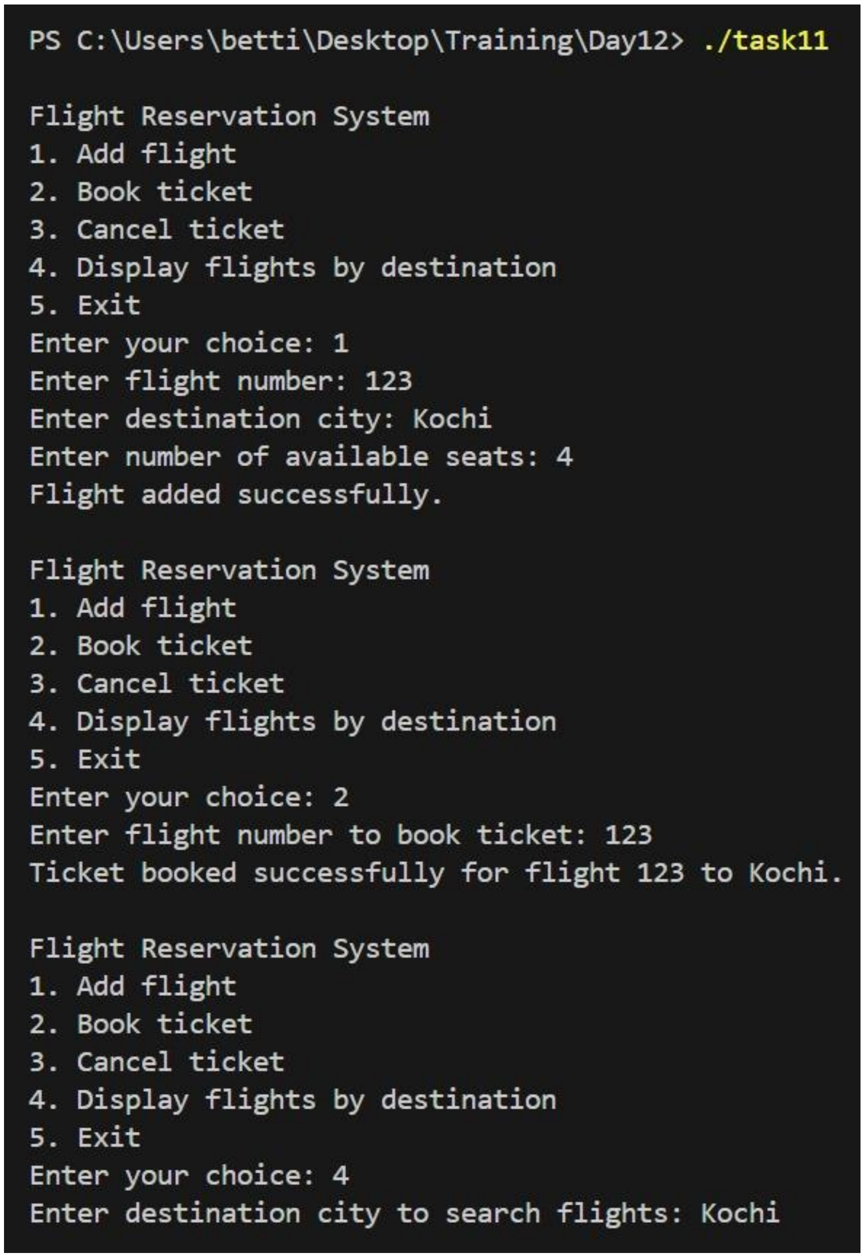


```
Enter your choice: 1

Enter details for student 1:
Roll Number: 1
Name: Bettina
Enter marks for 5 subjects: 10 29 30 45 16

Enter details for student 2:
Roll Number: 2
Name: Akash
Enter marks for 5 subjects: 25 35 46 50 38

1. Input student details
2. Calculate grades
3. Display student details with grades
4. Exit
Enter your choice: 2

1. Input student details
2. Calculate grades
3. Display student details with grades
4. Exit
Enter your choice: 3

Details of students with grades:
Roll No  Name                Marks                   Grade
------------------------------------------------------------
1        Bettina            10.00 29.00 30.00 45.00 16.00     F
2        Akash              25.00 35.00 46.00 50.00 38.00     F

1. Input student details
2. Calculate grades
3. Display student details with grades
4. Exit
Enter your choice: 4
Exiting...
```

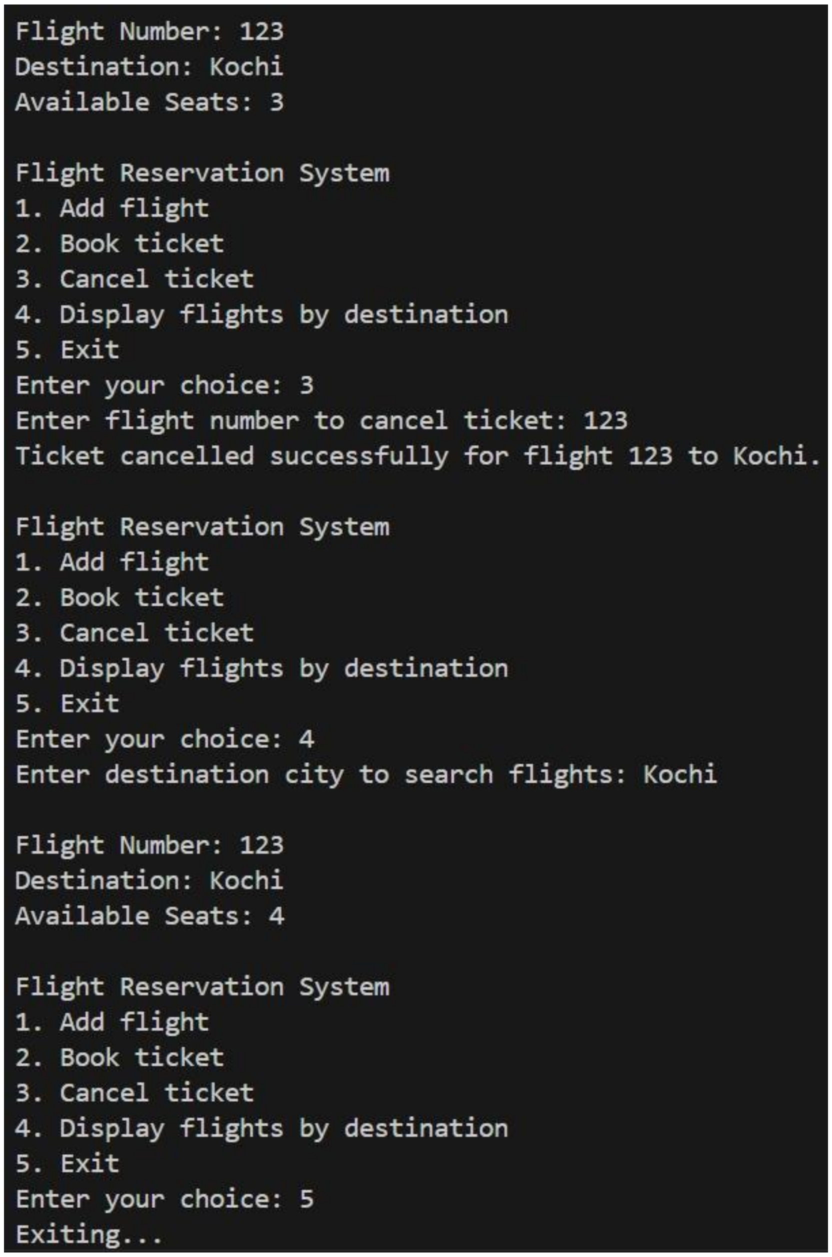
10.

```
/*Problem 5: Flight Reservation System
Objective: Simulate a simple flight reservation system using structures.
Description:
Define a structure Flight with fields:
char flight_number[10]: Flight number
char destination[50]: Destination city
int available_seats: Number of available seats
Write a program to:
```

```c
Add flights to the system.
Book tickets for a flight, reducing available seats accordingly.
Display the flight details based on destination.
Cancel tickets, increasing the number of available seats.*/

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

struct Flight {
    char flight_number[10];
    char destination[50];
    int available_seats;
};

void addFlight(struct Flight flights[], int *noOfFlights);
void bookTicket(struct Flight flights[], int noOfFlights);
void cancelTicket(struct Flight flights[], int noOfFlights);
void displayFlights(struct Flight flights[], int noOfFlights);

int main() {
    struct Flight flights[100];
    int noOfFlights = 0;
    int choice;

    while(1) {
        printf("\nFlight Reservation System\n");
        printf("1. Add flight\n");
        printf("2. Book ticket\n");
        printf("3. Cancel ticket\n");
        printf("4. Display flights by destination\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                addFlight(flights, &noOfFlights);
                break;
            case 2:
                bookTicket(flights, noOfFlights);
                break;
            case 3:
                cancelTicket(flights, noOfFlights);
                break;
            case 4:
                displayFlights(flights, noOfFlights);
                break;
            case 5:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void addFlight(struct Flight flights[], int *noOfFlights) {
    printf("Enter flight number: ");
    scanf("%s", flights[*noOfFlights].flight_number);

    printf("Enter destination city: ");
    scanf("%s", flights[*noOfFlights].destination);

    printf("Enter number of available seats: ");
    scanf("%d", &flights[*noOfFlights].available_seats);

    (*noOfFlights)++;
    printf("Flight added successfully.\n");
}

void bookTicket(struct Flight flights[], int noOfFlights) {
    char flightNumber[10];
    printf("Enter flight number to book ticket: ");
    scanf("%s", flightNumber);

    for(int i = 0; i < noOfFlights; i++) {
        if(strcmp(flights[i].flight_number, flightNumber) == 0) {
            if(flights[i].available_seats > 0) {
                flights[i].available_seats--;
                printf("Ticket booked successfully for flight %s to %s.\n", 
flights[i].flight_number, flights[i].destination);
                return;
            } else {
                printf("No available seats on this flight.\n");
                return;
            }
        }
    }
    printf("Flight not found.\n");
}

void cancelTicket(struct Flight flights[], int noOfFlights) {
    char flightNumber[10];
    printf("Enter flight number to cancel ticket: ");
    scanf("%s", flightNumber);

    for(int i = 0; i < noOfFlights; i++) {
        if(strcmp(flights[i].flight_number, flightNumber) == 0) {
            flights[i].available_seats++;
            printf("Ticket cancelled successfully for flight %s to %s.\n",
flights[i].flight_number, flights[i].destination);
            return;
        }
    }
    printf("Flight not found.\n");
}

void displayFlights(struct Flight flights[], int noOfFlights) {
    char destination[50];
    printf("Enter destination city to search flights: ");
    scanf("%s", destination);

    int found = 0;
    for (int i = 0; i < noOfFlights; i++) {
        if (strcmp(flights[i].destination, destination) == 0) {
            printf("\nFlight Number: %s\n", flights[i].flight_number);
            printf("Destination: %s\n", flights[i].destination);
            printf("Available Seats: %d\n", flights[i].available_seats);
            found = 1;
        }
    }

    if (!found) {
        printf("No flights available to %s.\n", destination);
    }
}
```



```
PS C:\Users\betti\Desktop\Training\Day12> ./task11

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 1
Enter flight number: 123
Enter destination city: Kochi
Enter number of available seats: 4
Flight added successfully.

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 2
Enter flight number to book ticket: 123
Ticket booked successfully for flight 123 to Kochi.

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 4
Enter destination city to search flights: Kochi
```



```
Flight Number: 123
Destination: Kochi
Available Seats: 3

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 3
Enter flight number to cancel ticket: 123
Ticket cancelled successfully for flight 123 to Kochi.

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 4
Enter destination city to search flights: Kochi

Flight Number: 123
Destination: Kochi
Available Seats: 4

Flight Reservation System
1. Add flight
2. Book ticket
3. Cancel ticket
4. Display flights by destination
5. Exit
Enter your choice: 5
Exiting...
```