Problem 1: Dynamic Array Resizing

Objective: Write a program to dynamically allocate an integer array and allow the user to resize it.

Description:

1. The program should ask the user to enter the initial size of the array.
2. Allocate memory using malloc.
3. Allow the user to enter elements into the array.
4. Provide an option to increase or decrease the size of the array. Use realloc to adjust the size.
5. Print the elements of the array after each resizing operation.

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int size,num1,num2,size2;
    char choice;
    printf("enter the number of elements of the array \n");
    scanf("%d",&size);
    int *ptr=(int *)malloc(size*sizeof(int));
    printf("enter the elements of the array \n");
    for(int i=0;i<size;i++)
    {
        scanf("%d",&num1);
        ptr[i]=num1;
    }
    printf("the elements of the array are \n");
    for(int i=0;i<size;i++)
    {
        printf("%d ",ptr[i]);

    }

    printf("\n");
    printf("do you want to enter more elements? y for y, n for no \n");
    scanf(" %c",&choice);
    switch(choice)
    {
        case 'y':
            printf("enter how many more elements you would like to add \n");
            scanf("%d",&size2);
            printf("enter the elements \n");
            ptr=(int *)realloc(ptr,size2);
            for(int i=0;i<size2;i++)
            {
                scanf("%d",&ptr[i]);
            }
            break;
        case 'n':
            break;
        default:
            printf("wrong choice");
            break;
    }

    printf("the elements of the array are \n");
    for(int i=0;i<size2+size;i++)
    {
        printf("%d ",ptr[i]);

    }
    free(ptr);
    ptr=NULL;
}
```

==

Problem 2: String Concatenation Using Dynamic Memory

Objective: Create a program that concatenates two strings using dynamic memory allocation.

Description:

1. Accept two strings from the user.
2. Use malloc to allocate memory for the first string.
3. Use realloc to resize the memory to accommodate the concatenated string.
4. Concatenate the strings and print the result.
5. Free the allocated memory.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char *str1, *str2, *result;
    int len1, len2;
    printf("Enter the first string: ");
    str1 = (char *)malloc(100 * sizeof(char));
    if (str1 == NULL)
    {
        printf("Memory allocation failed for the first string.\n");
        return 1;
    }
    fgets(str1, 100, stdin);
    str1[strcspn(str1, "\n")] = '\0';
    printf("Enter the second string: ");
    str2 = (char *)malloc(100 * sizeof(char));
    if (str2 == NULL) {
        printf("Memory allocation failed for the second string.\n");
        free(str1);
        return 1;
    }
    fgets(str2, 100, stdin);
    len1 = strlen(str1);
    len2 = strlen(str2);
    result = (char *)realloc(str1, (len1 + len2 + 1) * sizeof(char));
    if (result == NULL) {
        printf("Memory reallocation failed.\n");
        free(str1);
        free(str2);
        return 1;
    }
    strcat(result, str2);
    printf("The concatenated string is: %s\n", result);
    free(result);
    free(str2);
    return 0;
}

```

===

Problem 5: Dynamic 2D Array Allocation

Objective: Write a program to dynamically allocate a 2D array.

Description:

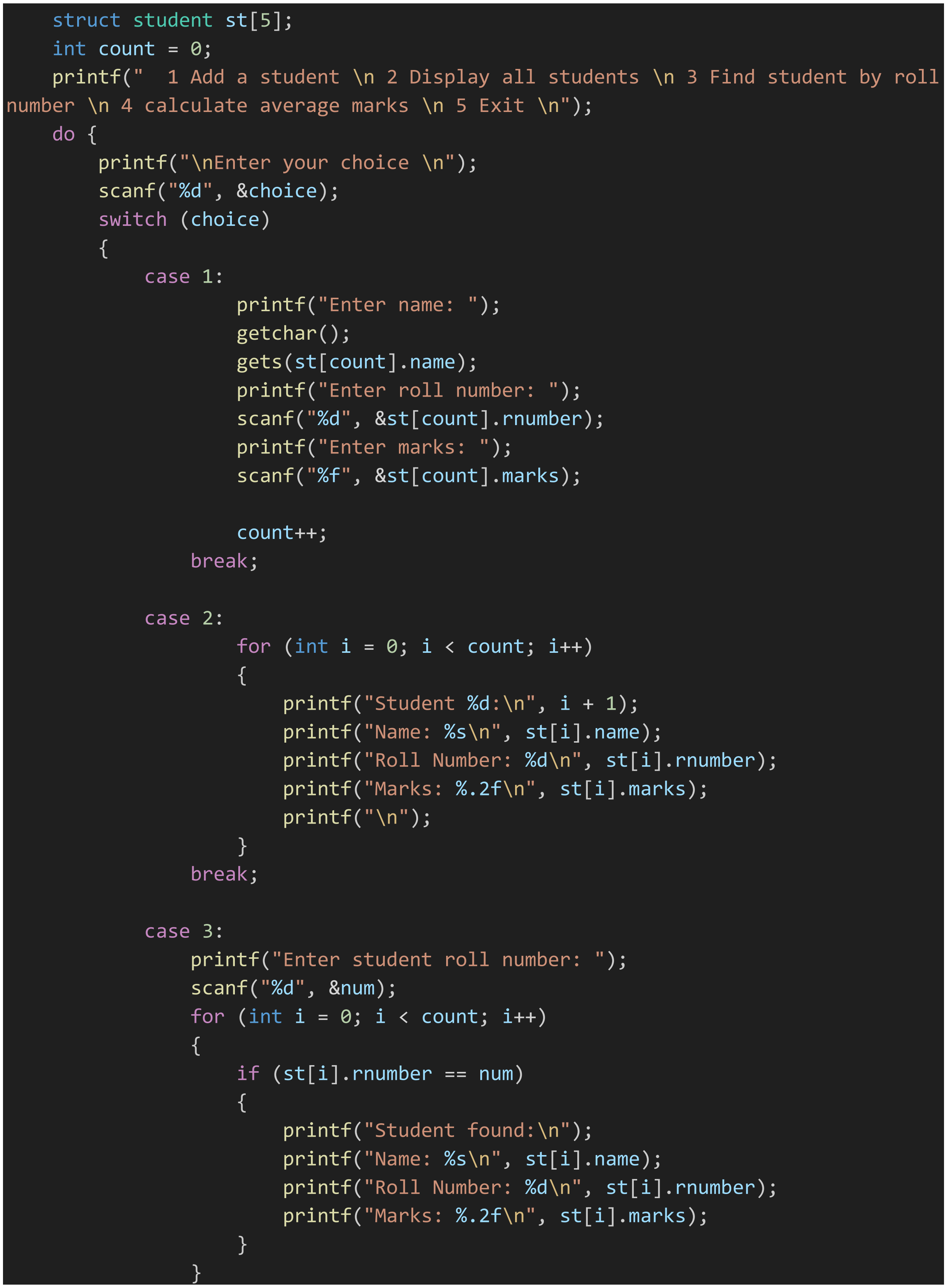
1. Accept the number of rows and columns from the user.
2. Use malloc (or calloc) to allocate memory for the rows and columns dynamically.
3. Allow the user to input values into the 2D array.
4. Print the array in matrix format.
5. Free all allocated memory at the end

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int rows, cols;
    int **matrix;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);
    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 the elements of the matrix:\n");
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            printf("Element [%d][%d]: ", i + 1, j + 1);
            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);
    printf("\nMemory has been freed successfully.\n");
    return 0;
}
```

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

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

int main() {
    int choice, num;
```



```c
    struct student st[5];
    int count = 0;
    printf("  1 Add a student \n 2 Display all students \n 3 Find student by roll
number \n 4 calculate average marks \n 5 Exit \n");
    do {
        printf("\nEnter your choice \n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                    printf("Enter name: ");
                    getchar();
                    gets(st[count].name);
                    printf("Enter roll number: ");
                    scanf("%d", &st[count].rnumber);
                    printf("Enter marks: ");
                    scanf("%f", &st[count].marks);

                    count++;
                break;

            case 2:
                    for (int i = 0; i < count; i++)
                    {
                        printf("Student %d:\n", i + 1);
                        printf("Name: %s\n", st[i].name);
                        printf("Roll Number: %d\n", st[i].rnumber);
                        printf("Marks: %.2f\n", st[i].marks);
                        printf("\n");
                    }
                break;

            case 3:
                printf("Enter student roll number: ");
                scanf("%d", &num);
                for (int i = 0; i < count; i++)
                {
                    if (st[i].rnumber == num)
                    {
                        printf("Student found:\n");
                        printf("Name: %s\n", st[i].name);
                        printf("Roll Number: %d\n", st[i].rnumber);
                        printf("Marks: %.2f\n", st[i].marks);
                    }
                }
                break;

            case 4:
                if (count == 0)
                {
                    printf("No students added yet. Cannot calculate average.\n");
                } else
                {
                    float avg = 0;
                    for (int i = 0; i < count; i++)
                    {
                        avg += st[i].marks;
                    }
                    avg /= count;
                    printf("Average marks: %.2f\n", avg);
                }
                break;

            case 5:
                printf("Exiting system.\n");
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (1);
    return 0;
}
```

Problem 1: Employee Management System

Objective: Create a program to manage employee details using structures.

Description:

1. Define a structure Employee with fields:
    - int emp_id: Employee ID
    - char name[50]: Employee name
    - float salary: Employee salary
2. Write a menu-driven program to:
    - Add an employee.
    - Update employee salary by ID.

- Display all employee details.
- Find and display details of the employee with the highest salary.

```c
 #include<stdio.h>

#include<stdlib.h>
int main()
{
    int choice,i,num;
     float highest=0;
    static int count=0;
    struct employee
    {
        int id;
        char name[50];
        float salary;
    };
    struct employee emp[5];
    printf("\nAdd an employee. \nUpdate employee salary by ID. \nDisplay all
employee details. \nFind and display details of the employee with the highest
salary.\n");
    do
    {
    printf("enter choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            printf("enter employee id \n");
            scanf("%d",&emp[count].id);
              getchar();
            printf("enter employee name \n");
            gets(emp[count].name);
            printf("enter salary \n ");
            scanf("%f",&emp[count].salary);
            count++;
            break;

        case 2:
            printf("enter employee id \n");
            scanf("%d",&num);
            for(int i=0;i<count;i++)
            {
                if(num==emp[i].id)
                {
                    printf("enter the new salary \n");
                    scanf("%f",&emp[i].salary);
                    break;
                }
                else
                    printf("employee not found \n");
            }
            break;

        case 3:
            printf("employee details \n");
            for(int i=0;i<count;i++)
            {
                printf("employee %d \n",i+1);
                printf("employee id= %d, employee name= %s, employee salary=
%0.2f \n",emp[i].id,emp[i].name,emp[i].salary);
            }
            break;
        case 4:
            for(int i=0;i<count;i++)
            {
                if(emp[i].salary>highest)
                    highest=emp[i].salary;
            }
            printf("the highest salary is %0.2f",highest);
            break;

        case 5:
            printf("exiting program \n");
            exit(0);

        default:
            printf("wrong choice ");
    }
    } while (1);
    return 0;
}
```

Problem 2: Library Management System

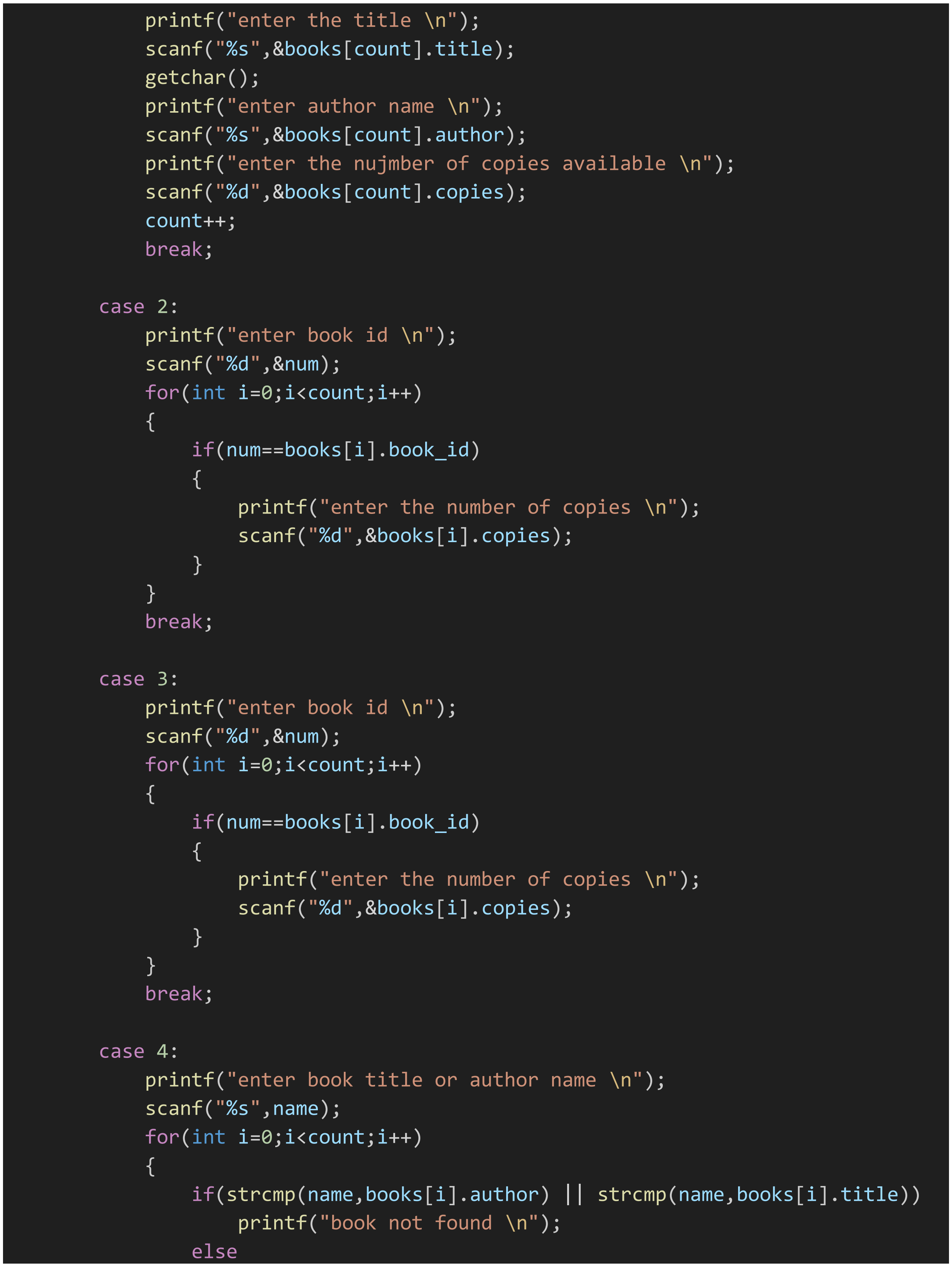
Objective: Manage a library system with a structure to store book details.

Description:

1. 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
2. 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.

```c
#include<stdio.h>

#include<stdlib.h>
#include<string.h>
int main()
{
    int choice,num,count=0;
    struct book
    {
        int book_id;
        char title[100];
        char author[50];
        int copies;
    };
    struct book books[5];
    char name[50];
    printf("Add books to the library.\nIssue a book by reducing the number of
copies.\nReturn a book by increasing the number of copies.\nSearch for a book by
title or author name.\n");
    do
    {
    printf("enter a choice \n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            printf("enter the book id \n");
            scanf("%d",&books[count].book_id);
            getchar();
```



```c
            printf("enter the title \n");
            scanf("%s",&books[count].title);
            getchar();
            printf("enter author name \n");
            scanf("%s",&books[count].author);
            printf("enter the nujmber of copies available \n");
            scanf("%d",&books[count].copies);
            count++;
            break;

        case 2:
            printf("enter book id \n");
            scanf("%d",&num);
            for(int i=0;i<count;i++)
            {
                if(num==books[i].book_id)
                {
                    printf("enter the number of copies \n");
                    scanf("%d",&books[i].copies);
                }
            }
            break;

        case 3:
            printf("enter book id \n");
            scanf("%d",&num);
            for(int i=0;i<count;i++)
            {
                if(num==books[i].book_id)
                {
                    printf("enter the number of copies \n");
                    scanf("%d",&books[i].copies);
                }
            }
            break;

        case 4:
            printf("enter book title or author name \n");
            scanf("%s",name);
            for(int i=0;i<count;i++)
            {
                if(strcmp(name,books[i].author) || strcmp(name,books[i].title))
                    printf("book not found \n");
                else
```

```
                    printf("book id= %d, title= %s, author= %s, number of copies=
%d \n",books[i].book_id,books[i].title,books[i].author,books[i].copies);
            }
            break;

        case 5:
            printf("exiting the system \n");
            exit(0);

        default :
            printf("wrong choice \n");
            break;
    }
    }while(1);
}
```

Problem 3: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. 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
2. 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 <stdlib.h>
#include <string.h>
struct Player
{
    char name[50];
    int matches;
    int runs;
    float average;
};
int main()
{
    struct Player *players = NULL;
    int n, choice, count = 0;
    int capacity = 0;
    printf("Menu:\n");
    printf("1. Add a player\n");
    printf("2. Display all players\n");
    printf("3. Find player with highest batting average\n");
    printf("4. Exit\n");
    do {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (count == capacity) {
                    if (capacity == 0) {
                        capacity = 1;
                    } else {
                        capacity *= 2;
                    }
                    players = (struct Player *)realloc(players, capacity *
sizeof(struct Player));
                    if (players == NULL) {
                        printf("Memory allocation failed. Exiting.\n");
                        exit(1);
                    }
                }
                printf("Enter player name: ");
                getchar();
                fgets(players[count].name, 50, stdin);
                players[count].name[strcspn(players[count].name, "\n")] = '\0';
                printf("Enter number of matches played: ");
                scanf("%d", &players[count].matches);
                printf("Enter total runs scored: ");
                scanf("%d", &players[count].runs);
                if (players[count].matches > 0) {
                    players[count].average = (float)players[count].runs /
players[count].matches;
                } else {
                    players[count].average = 0.0;
                }
                count++;
                break;

            case 2:
                if (count == 0)
                {
                    printf("No players added yet.\n");
                } else
                {
                    for (int i = 0; i < count; i++) {
                        printf("Player %d:\n", i + 1);
                        printf("Name: %s\n", players[i].name);
                        printf("Matches: %d\n", players[i].matches);
                        printf("Runs: %d\n", players[i].runs);
                        printf("Batting Average: %.2f\n", players[i].average);
                        printf("\n");
                    }
                }
                break;

            case 3:
                if (count == 0)
                {
                    printf("No players added yet.\n");
                } else
                {
                    int maxIndex = 0;
                    for (int i = 1; i < count; i++)
                    {
                        if (players[i].average > players[maxIndex].average)
                        {
                            maxIndex = i;
                        }
                    }
                    printf("Player with highest batting average:\n");
                    printf("Name: %s\n", players[maxIndex].name);
                    printf("Matches: %d\n", players[maxIndex].matches);
                    printf("Runs: %d\n", players[maxIndex].runs);
                    printf("Batting Average: %.2f\n", players[maxIndex].average);
                }
                break;

            case 4:
                printf("Exiting program.\n");
                free(players);
```

```
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (1);
    return 0;
}
```

Problem 4: Student Grading System

Objective: Manage student data and calculate grades based on marks.

Description:

1. Define a structure Student with fields:
   - int roll_no: Roll number
   - char name[50]: Student name
   - float marks[5]: Marks in 5 subjects
   - char grade: Grade based on the average marks
2. Write a program to:
   - Input details of n students.
   - Calculate the average marks and assign grades (A, B, C, etc.).
   - Display details of students along with their grades.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char grade(float avg);
struct Student
{
    int roll_no;
    char name[50];
    float marks[5];
    char grade;
};
```

```c
int main()
{
    struct Student *students = NULL;
    int n, count = 0, capacity = 0, choice;
    printf("Menu:\n");
    printf("1. Add a student\n");
    printf("2. Display all students\n");
    printf("3. Exit\n");
    do {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (count == capacity)
                {
                    if (capacity == 0)
                    {
                        capacity = 1;
                    } else {
                        capacity *= 2;
                    }
                    students = (struct Student *)realloc(students, capacity *
sizeof(struct Student));
                    if (students == NULL)
                     {
                        printf("Memory allocation failed. Exiting.\n");
                        exit(1);
                    }
                }
                printf("Enter roll number: ");
                scanf("%d", &students[count].roll_no);
                printf("Enter name: ");
                getchar();
                fgets(students[count].name, 50, stdin);
                printf("Enter marks in 5 subjects: ");
                float total = 0;
                for (int i = 0; i < 5; i++)
                {
                    scanf("%f", &students[count].marks[i]);
                    total += students[count].marks[i];
                }
                float average = total / 5.0;
                students[count].grade = grade(average);
                count++;
                break;

            case 2:
                if (count == 0)
                {
                    printf("No students to display.\n");
                } else
                {
                    for (int i = 0; i < count; i++)
                    {
                        printf("Student %d:\n", i + 1);
                        printf("Roll Number: %d\n", students[i].roll_no);
                        printf("Name: %s\n", students[i].name);
                        printf("Marks: ");
                        for (int j = 0; j < 5; j++)
                        {
                            printf("%.2f ", students[i].marks[j]);
                        }
                        printf("\nGrade: %c\n", students[i].grade);
                        printf("\n");
                    }
                }
                break;

            case 3:
                printf("Exiting program.\n");
                free(students);
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (1);
    return 0;
}
char grade(float avg)
{
    if (avg >= 90) return 'A';
    else if (avg >= 75) return 'B';
    else if (avg >= 60) return 'C';
    else if (avg >= 50) return 'D';
    else return 'F';
}
```

Problem 5: Flight Reservation System

Objective: Simulate a simple flight reservation system using structures.

Description:

1. Define a structure Flight with fields:
   - char flight_number[10]: Flight number
   - char destination[50]: Destination city
   - int available_seats: Number of available seats
2. Write a program to:
   - 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.

```c
#include <stdio.h>

#include <stdlib.h>
#include <string.h>
struct Flight
{
    char flight_number[10];
    char destination[50];
    int available_seats;
};

int main()
{
    struct Flight *flights = NULL;
    int n = 0, capacity = 0, choice;
    char search_destination[50], flight_no[10];
    int tickets;
    printf("Menu:\n");
    printf("1. Add a flight\n");
    printf("2. Book tickets for a flight\n");
    printf("3. Display flight details by destination\n");
    printf("4. Cancel tickets for a flight\n");
    printf("5. Exit\n");
    do {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (n == capacity) {
                    if (capacity == 0)
                    {
                        capacity = 1;
                    } else
                     {
                        capacity *= 2;
                    }
                    flights = (struct Flight *)realloc(flights, capacity *
sizeof(struct Flight));
                    if (flights == NULL)
                    {
                        printf("Memory allocation failed. Exiting.\n");
                        exit(1);
                    }
                }

                printf("Enter flight number: ");
                getchar();
                fgets(flights[n].flight_number, 10, stdin);
                printf("Enter destination: ");
                fgets(flights[n].destination, 50, stdin);
                printf("Enter number of available seats: ");
                scanf("%d", &flights[n].available_seats);
                n++;
                break;

            case 2:
                printf("Enter flight number to book tickets: ");
                getchar();
                fgets(flight_no, 10, stdin);
                printf("Enter number of tickets to book: ");
                scanf("%d", &tickets);
                for (int i = 0; i < n; i++) {
                    if (strcmp(flights[i].flight_number, flight_no) == 0) {
                        if (flights[i].available_seats >= tickets) {
                            flights[i].available_seats -= tickets;
                            printf("Booking successful! Remaining seats: %d\n",
flights[i].available_seats);
                        } else {
                            printf("Not enough seats available. Only %d seats
left.\n", flights[i].available_seats);
```

```
                        }
                        break;
                    }
                }
                break;

            case 3:
                printf("Enter destination to search for flights: ");
                getchar();
                fgets(search_destination, 50, stdin);
                printf("Flights to %s:\n", search_destination);
                for (int i = 0; i < n; i++) {
                    if (strcmp(flights[i].destination, search_destination) == 0)
{
                        printf("Flight Number: %s\n", flights[i].flight_number);
                        printf("Available Seats: %d\n\n",
flights[i].available_seats);
                    }
                }
                break;

            case 4:
                printf("Enter flight number to cancel tickets: ");
                getchar();
                fgets(flight_no, 10, stdin);
                printf("Enter number of tickets to cancel: ");
                scanf("%d", &tickets);

                for (int i = 0; i < n; i++) {
                    if (strcmp(flights[i].flight_number, flight_no) == 0) {
                        flights[i].available_seats += tickets;
                        printf("Cancellation successful! Updated seats: %d\n",
flights[i].available_seats);
                        break;
                    }
                }
                break;

            case 5:
                printf("Exiting system.\n");
                free(flights);
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (1);

    return 0;
}
```