Assignment

Calloc()-

```c
#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr=NULL;
    int n;
    printf("Enter the number integers that will be stored:");
    scanf("%d",&n);
    ptr=(int*)calloc(n,sizeof(int));
    for(int i=0;i<n;i++){
        printf("ptr[%d]=%d,",i,ptr[i]);
    }

}
```

Output
Enter the number integers that will be stored:5
ptr[0]=0,ptr[1]=0,ptr[2]=0,ptr[3]=0,ptr[4]=0,

Realloc()-

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

int main(){
    char *str;
    str=(char*)malloc(15);
    strcpy(str,"jason");
    printf("String before realloc = %s ,Address = %u\n",str,str);
    str=(char*)realloc(str,25);
    strcat(str,".com");
    printf("String after realloc = %s ,Address = %u\n",str,str);
    free(str);
    return(0);
}
```

Output (address can also be different)
String before realloc = jason ,Address = 11540816
String after realloc = jason.com ,Address = 11540816

Double Pointer and increasing pointers

```c
#include<stdio.h>

int main(){
    int **ipp;
    int ***ipp1;
    int i=4,j=5,k=6;
    int *ip1,*ip2;
    ip1=&i;
    ip2=&j;
    ipp=&ip1;
    printf("001 i = %d\n",*ip1);
    printf("002 i = %d\n",**ipp);
    **ipp=8;
    printf("003 i =%d\n",i);
    ipp1=&ipp;
    printf("004 i = %d\n",***ipp1);
}
```

Output:
001 i = 4
002 i = 4
003 i =8
004 i = 8

Changinig the double pointer value

```c
#include<stdio.h>

int main(){
    int **ipp;
    int ***ipp1;
    int i=4,j=5,k=6;
    printf("Address of i = %p\n",&i);
    printf("Address of j = %p\n",&j);
    int *ip1,*ip2;
    ip1=&i;
    ip2=&j;
    ipp=&ip1;
    printf("===Before Modification===\n");
    printf("001 i = %d\n",*ip1);
    printf("002 i = %d\n",**ipp);
    ipp=&ip2;
    printf("===After Modification===\n");
    printf("002 j = %d\n",**ipp);
    printf("003 j = %p\n",*ipp);
    return 0;
}
```

Output
Address of i = 0061FF14
Address of j = 0061FF10
===Before Modification===
001 i = 4
002 i = 4
===After Modification===
002 j = 5
003 j = 0061FF10

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 *array = NULL;
    int size, new_size, option;
```

```
    printf("Enter Initial Size of Array: ");
    scanf("%d", &size);
    array = (int *)malloc(size * sizeof(int));
    if (array == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    printf("Enter %d elements:\n", size);
    for (int i = 0; i < size; i++) {
        printf("Element %d: ", i + 1);
        scanf("%d", &array[i]);
    }
    printf("Initial array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
    while (1) {
        printf("\nChoose an option:\n");
        printf("1. Increase size\n");
        printf("2. Decrease size\n");
        printf("3. Exit\n");
        printf("Choose an option: ");
        scanf("%d", &option);
        if (option > 3) {
            printf("Invalid option.\n");
            continue;
        } else if (option == 3) {
            break;
        }
        printf("Enter the new size of the array: ");
        scanf("%d", &new_size);
        if (new_size <= 0) {
            printf("Size must be positive.\n");
            continue;
        }
        array = (int *)realloc(array, new_size * sizeof(int));
        if (array == NULL) {
            printf("Memory reallocation failed.\n");
            return 1;
        }
        if (new_size > size) {
            printf("Enter %d new elements:\n", new_size - size);
            for (int i = size; i < new_size; i++) {
                printf("Element %d: ", i + 1);
                scanf("%d", &array[i]);
            }
        }
        size = new_size;
        printf("Array after resizing: ");
        for (int i = 0; i < size; i++) {
            printf("%d ", array[i]);
        }
        printf("\n");
    }
    free(array);
    return 0;
}
```

Output

Enter the initial size of the array: 3

Enter 3 elements:

Element 1: 1

Element 2: 2

Element 3: 3

Initial array: 1 2 3

Choose an option:

1. Increase size

2. Decrease size

3. Exit

Choose an option: 1

Enter the new size of the array: 2

Array after resizing: 1 2

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 = NULL, *str2 = NULL;
    int len1, len2;
    str1 = (char *)malloc(50 * sizeof(char));
    if (str1 == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    printf("Enter first string: ");
    scanf("%s", str1);
    str2 = (char *)malloc(50 * sizeof(char));
    if (str2 == NULL) {
        printf("Memory allocation failed.\n");
        free(str1);
        return 1;
```

```
    }
    printf("Enter second string: ");
    scanf("%s", str2);
    len1 = strlen(str1);
    len2 = strlen(str2);
    str1 = (char *)realloc(str1, (len1 + len2 + 1) * sizeof(char));
    if (str1 == NULL) {
        printf("Memory reallocation failed.\n");
        free(str2);
        return 1;
    }
    strcat(str1, str2);
    printf("Concatenated string: %s\n", str1);
    free(str1);
    free(str2);
    return 0;
}
```

Enter first string: hai

Enter second string: hello

Concatenated string: haihello

Problem 3: Sparse Matrix Representation

Objective: Represent a sparse matrix using dynamic memory allocation.

Description:

1. Accept a matrix of size m×nm \times nm×n from the user.
2. Store only the non-zero elements in a dynamically allocated array of structures (with fields for row, column, and value).
3. Print the sparse matrix representation.
4. Free the allocated memory at the end.

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

typedef struct
{
    int row;
    int col;
    int val;
}s_matrix;
int main()
{
    int m, n, count=0;
    printf("Enter the number of rows and columns of the matrix: ");
    scanf("%d %d", &m, &n);

    int** matrix = (int**)malloc(m * sizeof(int *));
    for (int i = 0; i < m; i++)
    {
        matrix[i] = (int*)malloc(n * sizeof(int));
    }

    printf("Enter the elements of 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)
            {
                count++;
            }
        }
    }

    s_matrix *sparse_mat = (s_matrix *)malloc(count * sizeof(s_matrix));
    int k = 0;

    for(int i=0; i<m; i++)
    {
        for(int j=0; j<n; j++)
        {
            if(matrix[i][j] != 0)
            {
                sparse_mat[k].row = i;
                sparse_mat[k].col = j;
                sparse_mat[k].val = matrix[i][j];
                k++;
            }
        }
    }

    printf("\nSparse Matrix Representation:\n");
    printf("Row\tColumn\tValue\n");
    for (int i = 0; i < count; i++)
    {
        printf("%d\t%d\t%d\n", sparse_mat[i].row, sparse_mat[i].col, sparse_mat[i].val);
    }
    for (int i = 0; i < m; i++)
    {
        free(matrix[i]);
    }
    free(matrix);
    free(sparse_mat);
}
```

Enter the number of rows and columns of the matrix: 2

3

Enter the elements of the matrix:

1

2

3

4

5

6

Sparse Matrix Representation:

Row Column Value

```
0   0   1

0   1   2

0   2   3

1   0   4

1   1   5

1   2   6
```

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 **array;
    int rows, cols;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);
    array = (int **)malloc(rows * sizeof(int *));
    if (array == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }
    for (int i = 0; i < rows; i++) {
        array[i] = (int *)malloc(cols * sizeof(int));
        if (array[i] == NULL) {
            printf("Memory allocation failed for row %d!\n", i);
            return 1;
        }
    }
    printf("\nEnter the values for the 2D array:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("Element at [%d][%d]: ", i, j);
            scanf("%d", &array[i][j]);
        }
    }
    printf("\nThe 2D array (Matrix) is:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", array[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < rows; i++) {
```

```
        free(array[i]);
    }
    free(array);
    printf("Memory freed successfully.\n");
    return 0;
}
```

Output
Enter the number of rows: 3
Enter the number of columns: 3

Enter the values for the 2D array:
Element at [0][0]: 1
Element at [0][1]: 2
Element at [0][2]: 3
Element at [1][0]: 2
Element at [1][1]: 1
Element at [1][2]: 4
Element at [2][0]: 5
Element at [2][1]: 6
Element at [2][2]: 7

The 2D array (Matrix) is:
1 2 3
2 1 4
5 6 7
Memory freed successfully.

Structure

```
#include<stdio.h>
struct date{   //define a structure
    int day;
    int month;  //3(no. of elements in struct)*4(size of int)
    int year;
};
int main(){
    struct date today;  //declaring a structure
    printf("Size of today = %ld\n",sizeof(today));
    today.day=21;
    today.month=11;
    today.year=2024;
    printf("Today's Date is : %d-%d-%d",today.day,today.month,today.year);
    return 0;
}
```

Output

Size of today = 12

Today's Date is : 21-11-2024

Problem 1: Student Record Management System

Objective

Create a program to manage student records using structures.

Requirements

1. Define a Student structure with the following fields:
   - char name[50]
   - int rollNumber
   - float marks
2. Implement functions to:
   - Add a new student record.
   - Display all student records.
   - Find and display a student record by roll number.
   - Calculate and display the average marks of all students.
3. Implement a menu-driven interface to perform the above operations.

Output

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: John Doe
Enter roll number: 101
Enter marks: 85.5

Student added successfully!

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

struct student {
    char name[50];
    int rollNumber;
    float marks;
};
struct student students[100];
int stdCount = 0;
void addStudent();
void displayStudent();
void avgmarks();
void findStudentByRollNumber();
int main() {
    int choice;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Add Student\n");
        printf("2. Display All Students\n");
        printf("3. Find Student by Roll Number\n");
        printf("4. Calculate Average Marks\n");
        printf("5. Exit\n");
        printf("Enter your Choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                addStudent();
                break;
            case 2:
                displayStudent();
                break;
            case 3:
                findStudentByRollNumber();
                break;
            case 4:
                avgmarks();
                break;
            case 5:
                printf("Exiting program\n");
                return 0;
            default:
```

```
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
void addStudent() {
    printf("Enter Name: ");
    scanf(" %[^\n]", students[stdCount].name);
    printf("Enter Roll Number: ");
    scanf("%d", &students[stdCount].rollNumber);
    printf("Enter Marks: ");
    scanf("%f", &students[stdCount].marks);
    stdCount++;
    printf("Student added successfully!\n");
}
void displayStudent(){
    if(stdCount==0){
        printf("No Students Found");
    }
    else{
        for(int i=0;i<stdCount;i++){
            printf("%d.\t",i+1);
            printf("%s\n",students[i].name);
        }
    }
}
void avgmarks(){
    float total_marks=0.0;
    for(int i=0;i<stdCount;i++){
        total_marks+=students[i].marks;
    }
    float Average_Marks=total_marks/stdCount;
    printf("Average Marks : %.2f\n",Average_Marks);
}
void findStudentByRollNumber() {
    int rollNumber;
    printf("Enter Roll Number to search: ");
    scanf("%d", &rollNumber);
    for (int i = 0; i < stdCount; i++) {
        if (students[i].rollNumber == rollNumber) {
            printf("Student Found:\n");
            printf("Name: %s\n", students[i].name);
            printf("Roll Number: %d\n", students[i].rollNumber);
            printf("Marks: %.2f\n", students[i].marks);
            return;
        }
    }
    printf("No student found with Roll Number %d.\n", rollNumber);
}
```

Output

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: sneha
Enter Roll Number: 18

Enter Marks: 50
Student added successfully!

Assignment with Compound Literals

```c
#include<stdio.h>
struct  Coordinate{
    int x;
    int y;
};
void printCoorinate(struct Coordinate);
int main(){
    printCoorinate((struct Coordinate){5,6});
    // struct Coordinate pointA={5,6};
    // printCoorinate(pointA);
    return 0;}
void printCoorinate(struct Coordinate temp){
    printf("x = %d y = %d \n",temp.x,temp.y);
}
```

Output : x = 5 y = 6

Coordinates

```c
#include<stdio.h>
struct  Coordinate{
    int x;
    int y;
};
int main(){

    struct Coordinate Pnt[5];
    for(int i=0;i<5;i++){
        printf("Initialize the struct present in the %d index \n",i);
        scanf("%d %d",&Pnt[i].x,&Pnt[i].y);
        printf("\n");
    }
    for(int i=0;i<5;i++){
        printf("Display the coordinates at index %d is (%d,%d)\n",i,Pnt[i].x,Pnt[i].y);
        printf("\n");
    }
    return 0;
}
```

Output

Initialize the struct present in the 4 index
8
2

Display the coordinates at index 0 is (4,3)

Display the coordinates at index 1 is (9,7)

Display the coordinates at index 2 is (2,8)

Display the coordinates at index 3 is (9,5)

Display the coordinates at index 4 is (8,2)

```c
#include <stdio.h>

struct Month{
    int noOfDays;
```

```
    char name[3];
};

int main(){
    struct Month allMonths[12];
    for(int i = 0; i < 12; i++){
        printf("Enter The Month Name and the no. of days associated with that month");
        scanf("%s  %d", allMonths[i].name, &allMonths[i].noOfDays);
        printf("\n");
    }
    for(int j = 0; j < 12; j++){
        printf("Name of the Month = %s having %d \n",allMonths[j].name,allMonths[j].noOfDays);
    }
}
```

Output

Enter The Month Name and the no. of days associated with that monthnov
30

Enter The Month Name and the no. of days associated with that monthdec
31

Name of the Month = jan having 31
Name of the Month = feb having 28

Problem 1: Employee Management System

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

Description:

1. Define a structure Employee with fields:
    1. int emp_id: Employee ID
    2. char name[50]: Employee name
    3. float salary: Employee salary
2. Write a menu-driven program to:
    1. Add an employee.
    2. Update employee salary by ID.
    3. Display all employee details.
    4. 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;
};
struct Employee employees[100];
int empCount = 0;
void addEmployee();
void updateSalary();
void displayEmployees();
void findHighestSalary();
int main() {
    int choice;
```

```c
    while (1) {
        printf("\nMenu:\n");
        printf("1. Add Employee\n");
        printf("2. Update Employee Salary by ID\n");
        printf("3. Display All Employees\n");
        printf("4. Find Employee with Highest Salary\n");
        printf("5. Exit\n");
        printf("Enter your Choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice) {
            case 1:
                addEmployee();
                break;
            case 2:
                updateSalary();
                break;
            case 3:
                displayEmployees();
                break;
            case 4:
                findHighestSalary();
                break;
            case 5:
                printf("Exiting program.\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
void addEmployee() {
    printf("Enter Employee ID: ");
    scanf("%d", &employees[empCount].emp_id);
    getchar();
    printf("Enter Employee Name: ");
    scanf("%[^\n]", employees[empCount].name);
    getchar();
    printf("Enter Employee Salary: ");
    scanf("%f", &employees[empCount].salary);
    getchar();
    empCount++;
    printf("Employee added successfully!\n");
}
void updateSalary() {
    int id;
    float newSalary;
    printf("Enter Employee ID to Update Salary: ");
    scanf("%d", &id);
    getchar();
    for (int i = 0; i < empCount; i++) {
        if (employees[i].emp_id == id) {
            printf("Enter New Salary for Employee %s: ", employees[i].name);
            scanf("%f", &newSalary);
            getchar();
            employees[i].salary = newSalary;
            printf("Salary updated successfully!\n");
            return;
        }
    }
```

```
    printf("Employee ID not found!\n");
}
void displayEmployees() {
    if (empCount == 0) {
        printf("No employees to display.\n");
        return;
    }
    printf("\nEmployee Details:\n");
    for (int i = 0; i < empCount; i++) {
        printf("ID: %d, Name: %s, Salary: %.2f\n",
               employees[i].emp_id, employees[i].name, employees[i].salary);
    }
}
void findHighestSalary() {
    if (empCount == 0) {
        printf("No employees to search.\n");
        return;
    }
    int highestIndex = 0;
    for (int i = 1; i < empCount; i++) {
        if (employees[i].salary > employees[highestIndex].salary) {
            highestIndex = i;
        }
    }
    printf("\nEmployee with the Highest Salary:\n");
    printf("ID: %d, Name: %s, Salary: %.2f\n",
           employees[highestIndex].emp_id, employees[highestIndex].name,
employees[highestIndex].salary);
}
```

Menu:
1. Add Employee
1. Add Employee
2. Update Employee Salary by ID
3. Display All Employees
4. Find Employee with Highest Salary
5. Exit
Enter your Choice: 2
Enter Employee ID to Update Salary: 1
Enter New Salary for Employee soumya: 25000
Salary updated successfully!

Menu:
1. Add Employee
2. Update Employee Salary by ID
3. Display All Employees
4. Find Employee with Highest Salary
5. Exit
Enter your Choice: 1
Enter Employee ID: 2
Enter Employee Name: sneha
Enter Employee Salary: 100000
Employee added successfully!

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:
    1. int book_id: Book ID
    2. char title[100]: Book title
    3. char author[50]: Author name
    4. int copies: Number of available copies
2. Write a program to:
    1. Add books to the library.
    2. Issue a book by reducing the number of copies.
    3. Return a book by increasing the number of copies.
    4. Search for a book by title or author name.

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

struct Book {
    int book_id;
    char title[100];
    char author[50];
    int copies;
};
struct Book library[100];
int bookCount = 0;
void addBooks();
void issueBook();
void returnBook();
void searchBook();
int main() {
    int choice;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Add Books\n");
        printf("2. Issue Books\n");
        printf("3. Return Books\n");
        printf("4. Search Books\n");
        printf("5. Exit\n");
        printf("Enter your Choice: ");
        scanf("%d", &choice);
        while (getchar() != '\n');
        switch (choice) {
            case 1:
                addBooks();
                break;
            case 2:
                issueBook();
                break;
            case 3:
                returnBook();
                break;
            case 4:
                searchBook();
                break;
            case 5:
                printf("Exiting program\n");
                return 0;
```

```
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
void addBooks() {
    printf("Enter Book ID: ");
    scanf("%d", &library[bookCount].book_id);
    while (getchar() != '\n');
    printf("Enter Book Title: ");
    scanf("%[^\n]", library[bookCount].title);
    while (getchar() != '\n');
    printf("Enter Author Name: ");
    scanf("%[^\n]", library[bookCount].author);
    while (getchar() != '\n');
    printf("Enter Number of Copies: ");
    scanf("%d", &library[bookCount].copies);
    while (getchar() != '\n');
    bookCount++;
    printf("Book added successfully!\n");
}
void issueBook() {
    char title[100];
    printf("Enter Title of the Book to Issue: ");
    scanf("%[^\n]", title);
    while (getchar() != '\n');
    for (int i = 0; i < bookCount; i++) {
        if (strcmp(library[i].title, title) == 0) {
            if (library[i].copies > 0) {
                library[i].copies--;
                printf("Book issued successfully! Remaining copies: %d\n", library[i].copies);
            } else {
                printf("Sorry, no copies available.\n");
            }
            return;
        }
    }
    printf("Book not found!\n");
}
void returnBook() {
    char title[100];
    printf("Enter Title of the Book to Return: ");
    scanf("%[^\n]", title);
    while (getchar() != '\n');
    for (int i = 0; i < bookCount; i++) {
        if (strcmp(library[i].title, title) == 0) {
            library[i].copies++;
            printf("Book returned successfully! Total copies: %d\n", library[i].copies);
            return;
        }
    }
    printf("Book not found!\n");
}
void searchBook() {
    char query[100];
    printf("Enter Title or Author to Search: ");
    scanf("%[^\n]", query);
    while (getchar() != '\n');
    for (int i = 0; i < bookCount; i++) {
```

```
        if (strstr(library[i].title, query) != NULL || strstr(library[i].author, query) !=
NULL) {
            printf("\nBook ID: %d\nTitle: %s\nAuthor: %s\nCopies: %d\n",
                   library[i].book_id, library[i].title, library[i].author, library[i].copies);
        }
    }
}
```

Output:
Menu:
1. Add Books
2. Issue Books
3. Return Books
4. Search Books
5. Exit
Enter your Choice: 1
Enter Book ID: 1
Enter Book Title: Wings of Fire
Enter Author Name: A P J Abdul Kalam
Enter Number of Copies: 5
Book added successfully!

Problem 3: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. Define a structure Player with fields:
    1. char name[50]: Player name
    2. int matches: Number of matches played
    3. int runs: Total runs scored
    4. float average: Batting average
2. Write a program to:
    1. Input details for n players.
    2. Calculate and display the batting average for each player.
    3. 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 inputPlayers(struct Player players[], int n);
void calculateAverage(struct Player players[], int n);
void findHighestAverage(struct Player players[], int n);
int main() {
    int n;
    printf("Enter the number of players: ");
```

```c
    scanf("%d", &n);
    getchar();
    struct Player players[n];
    inputPlayers(players, n);
    calculateAverage(players, n);
    findHighestAverage(players, n);
    return 0;
}
void inputPlayers(struct Player players[], int n) {
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for Player %d:\n", i + 1);
        printf("Enter Player Name: ");
        scanf("%[^\n]", players[i].name);
        getchar();
        printf("Enter Number of Matches Played: ");
        scanf("%d", &players[i].matches);
        getchar();
        printf("Enter Total Runs Scored: ");
        scanf("%d", &players[i].runs);
        getchar();
        players[i].average = 0.0;
    }
}
void calculateAverage(struct Player players[], int n) {
    printf("\nPlayer Averages:\n");
    for (int i = 0; i < n; i++) {
        if (players[i].matches > 0) {
            players[i].average = (float)players[i].runs / players[i].matches;
        } else {
            players[i].average = 0.0;
        }
        printf("Player: %s, Matches: %d, Runs: %d, Batting Average: %.2f\n",
               players[i].name, players[i].matches, players[i].runs, players[i].average);
    }
}
void findHighestAverage(struct Player players[], int n) {
    int highestIndex = 0;
    for (int i = 1; i < n; i++) {
        if (players[i].average > players[highestIndex].average) {
            highestIndex = i;
        }
    }
    printf("\nPlayer with the Highest Batting Average:\n");
    printf("Name: %s, Matches: %d, Runs: %d, Batting Average: %.2f\n",
           players[highestIndex].name, players[highestIndex].matches,
           players[highestIndex].runs, players[highestIndex].average);
}
```

Enter the number of players: 5

Enter details for Player 1:
Enter Player Name: soumya
Enter Number of Matches Played: 2
Enter Total Runs Scored: 150

Problem 4: Student Grading System

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

Description:

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

```c
#include <stdio.h>

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

```c
void inputStudents(struct Student students[], int *n);
void calculateGrades(struct Student students[], int n);
void displayStudents(struct Student students[], int n);
char assignGrade(float avg);
int main() {
    struct Student students[100];
    int n = 0;
    int choice;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Input Student Details\n");
        printf("2. Calculate Grades\n");
        printf("3. Display All Students\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice) {
            case 1:
                inputStudents(students, &n);
                break;
            case 2:
                calculateGrades(students, n);
                break;
            case 3:
                displayStudents(students, n);
                break;
            case 4:
                printf("Exiting program.\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
```

```c
void inputStudents(struct Student students[], int *n) {
    int num;
    printf("Enter the number of students to add: ");
    scanf("%d", &num);
    getchar();
    for (int i = 0; i < num; i++) {
        printf("\nEnter details for Student %d:\n", *n + 1);
        printf("Enter Roll Number: ");
        scanf("%d", &students[*n].roll_no);
        getchar();
        printf("Enter Name: ");
        scanf("%[^\n]", students[*n].name);
        getchar();
        printf("Enter Marks for 5 Subjects: ");
        for (int j = 0; j < 5; j++) {
            scanf("%f", &students[*n].marks[j]);
        }
        getchar();
        students[*n].grade = 'N';
        (*n)++;
    }
    printf("Student details added successfully!\n");
}
```

```c
void calculateGrades(struct Student students[], int n) {
    if (n == 0) {
        printf("No students available to calculate grades.\n");
        return;
    }
    for (int i = 0; i < n; i++) {
        float total = 0;
        for (int j = 0; j < 5; j++) {
            total += students[i].marks[j];
        }
        float avg = total / 5.0;
        students[i].grade = assignGrade(avg);
    }
    printf("Grades calculated successfully!\n");
}
char assignGrade(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';
}
void displayStudents(struct Student students[], int n) {
    if (n == 0) {
        printf("No students available to display.\n");
        return;
    }
    printf("\nStudent Details:\n");
    for (int i = 0; i < n; i++) {
        printf("Roll No: %d, Name: %s, Marks: ", students[i].roll_no, students[i].name);
        for (int j = 0; j < 5; j++) {
            printf("%.2f ", students[i].marks[j]);
```

```
        }
        printf(", Grade: %c\n", students[i].grade);
    }
}
```

Output

Menu:
1. Input Student Details
2. Calculate Grades
3. Display All Students
4. Exit
Enter your choice: 2
Grades calculated successfully!

Menu:
1. Input Student Details
2. Calculate Grades
3. Display All Students
4. Exit
Enter your choice: 3

Student Details:
Roll No: 1, Name: soumya, Marks: 25.00 30.00 45.00 65.00 45.00 , Grade: F
Roll No: 2, Name: sneha, Marks: 26.00 45.00 98.00 56.00 70.00 , Grade: D

Problem 5: Flight Reservation System

Objective: Simulate a simple flight reservation system using structures.

Description:

1. Define a structure Flight with fields:
    1. char flight_number[10]: Flight number
    2. char destination[50]: Destination city
    3. int available_seats: Number of available seats
2. Write a program to:
    1. Add flights to the system.
    2. Book tickets for a flight, reducing available seats accordingly.
    3. Display the flight details based on destination.
    4. 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 *flightCount);
void bookTicket(struct Flight flights[], int flightCount);
void displayFlightDetails(struct Flight flights[], int flightCount);
void cancelTicket(struct Flight flights[], int flightCount);
int main() {
```

```c
    struct Flight flights[100];
    int flightCount = 0;
    int choice;
    while (1) {
        printf("\nFlight Reservation System Menu:\n");
        printf("1. Add Flight\n");
        printf("2. Book Ticket\n");
        printf("3. Display Flight Details by Destination\n");
        printf("4. Cancel Ticket\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice) {
            case 1:
                addFlight(flights, &flightCount);
                break;
            case 2:
                bookTicket(flights, flightCount);
                break;
            case 3:
                displayFlightDetails(flights, flightCount);
                break;
            case 4:
                cancelTicket(flights, flightCount);
                break;
            case 5:
                printf("Exiting system.\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
void addFlight(struct Flight flights[], int *flightCount) {
    printf("\nEnter Flight Number: ");
    scanf("%s", flights[*flightCount].flight_number);
    getchar();
    printf("Enter Destination: ");
    scanf("%[^\n]", flights[*flightCount].destination);
    getchar();
    printf("Enter Available Seats: ");
    scanf("%d", &flights[*flightCount].available_seats);
    getchar();
    (*flightCount)++;
    printf("Flight added successfully!\n");
}
void bookTicket(struct Flight flights[], int flightCount) {
    char flight_number[10];
    int seats_to_book;
    printf("\nEnter Flight Number to Book Ticket: ");
    scanf("%s", flight_number);
    getchar();
    for (int i = 0; i < flightCount; i++) {
        if (strcmp(flights[i].flight_number, flight_number) == 0) {
            printf("Enter number of seats to book: ");
            scanf("%d", &seats_to_book);
            getchar();
            if (flights[i].available_seats >= seats_to_book) {
                flights[i].available_seats -= seats_to_book;
```

```
                printf("Booking successful! %d seats booked.\n", seats_to_book);
                printf("Remaining seats: %d\n", flights[i].available_seats);
                return;
            } else {
                printf("Not enough available seats. Only %d seats left.\n",
flights[i].available_seats);
                return;
            }
        }
    }
    printf("Flight with the given flight number not found!\n");
}
void displayFlightDetails(struct Flight flights[], int flightCount) {
    char destination[50];
    int found = 0;
    printf("\nEnter destination to search for flights: ");
    scanf("%[^\n]", destination);
    getchar();
    for (int i = 0; i < flightCount; 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 found for the destination '%s'.\n", destination);
    }
}
void cancelTicket(struct Flight flights[], int flightCount) {
    char flight_number[10];
    int seats_to_cancel;
    printf("\nEnter Flight Number to Cancel Ticket: ");
    scanf("%s", flight_number);
    getchar();
    for (int i = 0; i < flightCount; i++) {
        if (strcmp(flights[i].flight_number, flight_number) == 0) {
            printf("Enter number of seats to cancel: ");
            scanf("%d", &seats_to_cancel);
            getchar();
            if (seats_to_cancel > 0) {
                flights[i].available_seats += seats_to_cancel;
                printf("Cancellation successful! %d seats canceled.\n", seats_to_cancel);
                printf("Available seats: %d\n", flights[i].available_seats);
                return;
            } else {
                printf("Invalid seat number to cancel.\n");
                return;
            }
        }
    }
    printf("Flight with the given flight number not found!\n");
}
```

Output
Enter Flight Number to Book Ticket: 123
Enter number of seats to book: 2
Booking successful! 2 seats booked.
Remaining seats: 3

Flight Reservation System Menu:
Booking successful! 2 seats booked.
Remaining seats: 3

Flight Reservation System Menu:
Remaining seats: 3