calloc()

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *ptr = NULL;
   int n;
   printf("Enter the number of integers that wiil 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]);
   }
   return 0;

}
```

realloc()

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
   char *str = NULL;
   str = (char*)malloc(15);
   strcpy(str,"Jason");
   printf("String = %s, Address = %u\n",str,str);
   //Reallocating memory
   str = (char*)realloc(str,25);
   strcat(str,".com");
   printf("String = %s , Address = %u\n",str,str);
   free(str);
   return 0; }
```

DOUBLE POINTERS

```
#include <stdio.h>
int main() {
   int **ipp;
   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);
   return 0;

}
```

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.

```
 #include <stdio.h>
#include<stdlib.h>
int main() {
   int *ptr = NULL;
   int n,m;
   printf("Enter the initial size of the array: ");
   scanf("%d",&n);
```

```
    ptr = (int *)malloc(n*sizeof(int));
    for(int i=0;i<n;i++){
        scanf("%d",(ptr+i));
    }
    printf("Before resizing: \n");
    for(int i=0;i<n;i++){
        printf("ptr[%d] = %d, Address = %u\n",i,*(ptr+i),(ptr+i));
    }
    printf("Enter the new size of array: \n");
    scanf("%d",&m);

    ptr = (int *)realloc(ptr,m*sizeof(int));
    for(int i=0;i<m;i++){
        scanf("%d",(ptr+i));
    }
    printf("After resizing: \n");
    for(int i=0;i<m;i++){
        printf("ptr[%d] = %d, Address = %u\n",i,*(ptr+i),(ptr+i));
    }

    free(ptr);
}
```

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.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(){
   char str1[50],str2[100];
   printf("Enter str1: ");
   fgets(str1,sizeof(str1),stdin);
   str1[strcspn(str1, "\n")] = '\0';
   printf("Enter str2: ");
   fgets(str2,sizeof(str2),stdin);
   str1[strcspn(str1, "\n")] = '\0';
   char *pstr = NULL;
   pstr = (char *)malloc(50);
   strcat(str1,str2);
   pstr =(char *)realloc(pstr,100);
   printf("Concatenated string is %s",str1);
   free(pstr);
}
```

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.

Problem 4: Dynamic Linked List Implementation

Objective: Implement a linked list using dynamic memory allocation.

Description:

1. Define a struct for linked list nodes. Each node should store an integer and a pointer to the next node.
2. Create a menu-driven program to perform the following operations:
    - Add a node to the list.
    - Delete a node from the list.
    - Display the list.
3. Use malloc to allocate memory for each new node and free to deallocate memory for deleted nodes.

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.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
   int row,col;
   printf("Enter the values for row and column: ");
   scanf("%d %d",&row,&col);
   int matrix[row][col];
   int *ptr1 = NULL;
   int *ptr2 = NULL;
   ptr1 = (int *)malloc(row*sizeof(int));
   ptr2 = (int *)malloc(col*sizeof(int));
   for(int i=0;i<row;i++){
      for(int j=0;j<col;j++){
```

```
            scanf("%d",&matrix[i][j]);
        }
    }
     for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
    free(ptr1);
    free(ptr2);

    return 0;

}
```

STRUCTURE

```
#include<stdio.h>
struct date{
    int day;
    int month;
    int year;
}today;
int main(){
    //struct date today;
    today.day = 21;
    today.month = 11;
    today.year = 2024;
    printf("Today's date is %d/%d/%d\n",today.day,today.month,today.year);
    printf("size of today = %ld",sizeof(today));
    return 0;
```

```
        }
```

Un named structure

```
#include<stdio.h>
struct{
   int day;
   int month;
   int year;
}today,tomorrow;
int main(){
   //struct date today;
   today.day = 21;
   today.month = 11;
   today.year = 2024;
   printf("Today's date is %d/%d/%d\n",today.day,today.month,today.year);
   printf("size of today = %ld\n",sizeof(today));
   tomorrow.day = 22;
   tomorrow.month = 11;
   tomorrow.year = 2024;
   printf("Tomorrow's date is %d/%d/%d\n",tomorrow.day,tomorrow.month,tomorrow.year);
   return 0;
}
```

Initializing structure

```
#include<stdio.h>
struct date{
   int day;
   int month;
   int year;
};
int main(){
   struct date today={21,11,2024};
   printf("Today's date is %d/%d/%d\n",today.day,today.month,today.year);
```

```
    printf("size of today = %ld\n",sizeof(today));
    return 0;
}
#include <stdio.h>
struct student{
    char name[50];
    int rollNumber;
    float marks;
};
int main()
{
    struct student s1 = {.marks = 95.5,.rollNumber=7};
    printf("S1's Roll number and mark are %d & %f",s1.rollNumber,s1.marks);

    return 0;
}
```

Student structure

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
struct Student {
    char name[50];
    int rollNumber;
    float marks;
};
void addrecord(struct Student students[], int *studentCount);
void displayrecords(struct Student students[], int studentCount);
void findRecordByRollNumber(struct Student students[], int studentCount, int rollNumber);
void averageMarks(struct Student students[], int studentCount);

int main() {
```

```
struct Student students[100];
int studentCount = 0;
int choice, rollNumber;
printf("\n1. Add Student\n");
printf("2. Display All Students\n");
printf("3. Find Student by Roll Number\n");
printf("4. Calculate Average Marks\n");
printf("S Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice) {
   case 1:
      addrecord(students, &studentCount);
      break;
   case 2:
      displayrecords(students, studentCount);
      break;
   case 3:
      printf("Enter roll number to search: ");
      scanf("%d", &rollNumber);
      findRecordByRollNumber(students, studentCount, rollNumber);
      break;
   case 4:
      averageMarks(students, studentCount);
      break;
   case 5:
      printf("Exiting the program.\n");
      exit(0);
      break;
   default:
      printf("Invalid choice! Please try again.\n");
```

```
    }

    return 0;
}
void addrecord(struct Student students[], int *studentCount) {
    if (*studentCount >= 100) {
        printf("Student limit reached! Cannot add more students.\n");
        return;
    }
    struct Student newStudent;
    printf("Enter name: ");
    getchar();
    fgets(newStudent.name, sizeof(newStudent.name), stdin);
    newStudent.name[strcspn(newStudent.name, "\n")] = 0;
    printf("Enter roll number: ");
    scanf("%d", &newStudent.rollNumber);
    printf("Enter marks: ");
    scanf("%f", &newStudent.marks);
    students[*studentCount] = newStudent;
    (*studentCount)++;
    printf("Student added successfully!\n");
}
void displayrecords(struct Student students[], int studentCount) {
    if (studentCount == 0) {
        printf("No student records available.\n");
        return;
    }

    printf("\nStudent Records:\n");
    for (int i = 0; i < studentCount; i++) {
```

```
        printf("Name: %s, Roll Number: %d, Marks: %.2f\n", students[i].name, students[i].rollNumber,
students[i].marks);
    }
}
void findRecordByRollNumber(struct Student students[], int studentCount, int rollNumber) {
    int found = 0;

    for (int i = 0; i < studentCount; i++) {
        if (students[i].rollNumber == rollNumber) {
            printf("Student found:\n");
            printf("Name: %s, Roll Number: %d, Marks: %.2f\n", students[i].name, students[i].rollNumber,
students[i].marks);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Student with roll number %d not found.\n", rollNumber);
    }
}
void averageMarks(struct Student students[], int studentCount) {
    if (studentCount == 0) {
        printf("No student records available to calculate average marks.\n");
        return;
    }

    float totalMarks = 0;
    for (int i = 0; i < studentCount; i++) {
        totalMarks += students[i].marks;
    }
```

```
    float average = totalMarks / studentCount;
    printf("Average marks of all students: %.2f\n", average);
}
```

Compound literals

```
#include <stdio.h>
struct coordinate{
    int x;
    int y;
};
void printCoordinate(struct coordinate);
int main()
{
    printCoordinate((struct coordinate){5,6});
    /*struct coordinate pointA = {5,6};
    printCoordinate(pointA);*/

    return 0;
}
void printCoordinate(struct coordinate temp){
    printf("x = %d y = %d",temp.x,temp.y);
}
```

Array of structures

```
#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 structure 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 %d index is (%d,%d)\n",i,pnt[i].x,pnt[i].y);
    }


    return 0;
}


#include <stdio.h>
struct date{
    int day;
    int month;
    int year;
};
int main()
{
   struct date myDates[5]={{12,10,1975},{12,3,1980},{15,11,2005}};
   printf("Set first three dates in myDate to %d/%d/%d,%d/%d/%d,and
%d/%d/%d\n",myDates[0].day,myDates[0].month,myDates[0].year,myDates[1].day,myDates[1].mont
h,myDates[1].year,myDates[2].day,myDates[2].month,myDates[2].year);
   //struct date myDates[5]={12,10,1975,12,3,1980,15,11,2005};
   //printf("Set first three dates in myDate to %d/%d/%d,%d/%d/%d,and
%d/%d/%d\n",myDates[0].day,myDates[0].month,myDates[0].year,myDates[1].day,myDates[1].mont
h,myDates[1].year,myDates[2].day,myDates[2].month,myDates[2].year);
 return 0;
}
```

Structures containing arrays

```
#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("%s have %d days\n",allMonths[j].name,allMonths[j].noOfDays);
  }

  return 0;
}
```

Output:

```
Enter the month name and the no.of days associated with that month: Jan 31

Enter the month name and the no.of days associated with that month: Feb 28

Enter the month name and the no.of days associated with that month: Mar 31

Enter the month name and the no.of days associated with that month: Apr 30

Enter the month name and the no.of days associated with that month: May 31

Enter the month name and the no.of days associated with that month: Jun 30
```

Enter the month name and the no.of days associated with that month: Jul 31

Enter the month name and the no.of days associated with that month: Aug 31

Enter the month name and the no.of days associated with that month: Sep 30

Enter the month name and the no.of days associated with that month: Oct 31

Enter the month name and the no.of days associated with that month: Nov 30

Enter the month name and the no.of days associated with that month: Dec 31

Jan have 31 days

Feb have 28 days

Mar have 31 days

Apr have 30 days

May have 31 days

Jun have 30 days

Jul have 31 days

Aug have 31 days

Sep have 30 days

Oct have 31 days

Nov have 30 days

Dec have 31 days

Nested structures

```
#include <stdio.h>
struct currentDate{
   int day;
   int month;
```

```
    int year;
};
struct currentTime{
    int sec;
    int min;
    int hours;
};
struct cDateTime{
    struct currentDate d1;
    struct currentTime t1;
};
int main()
{
    struct cDateTime dt = {{21,11,2024},{51,01,17}};
    printf("Current Date = %d/%d/%d \n",dt.d1.day,dt.d1.month,dt.d1.year);
    printf("Current Time = %d:%d:%d \n",dt.t1.sec,dt.t1.min,dt.t1.hours);


  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.

```
#include<stdio.h>
#include<string.h>
struct Employee{
   int emp_id;
   char name[50];
   float salary;
};
void add(struct Employee employees[],int *count);
void updateSalary(struct Employee employees[],float salary,int id,int count);
void display(struct Employee employees[],int count);
void find(struct Employee employees[],int count);
int main(){
   struct Employee employees[50];
   int choice,count=0,id,op;
   float salary;
   char cont;
   do{
   printf("1.Add an employee\n2.Update employee salary by ID\n3.Display all employee
details\n4.Find and display details of employee with the highest salary\n");
   printf("Enter the choice: ");
   scanf(" %d",&op);
   switch(op){
      case 1:
         add(employees,&count);
         break;
      case 2:
         printf("Enter the new salary: \n");
         scanf("%f",&salary);
         printf("Enter the id of employee whose salary is to be updated: \n");
         scanf("%d",&id);
```

```
            updateSalary(employees,salary,id,count);
            break;
        case 3:
            display(employees,count);
            break;
        default:
            printf("Invalid option!");

    }
    printf("\nDo you want to continue? (y/n): ");
    getchar();
    scanf("%c", &cont);
    }while (cont == 'y' || cont == 'Y');
}
void add(struct Employee employees[],int *count){
    if(*count>50){
        printf("Cannot add new employee");
    }else{
        printf("Enter employee ID: ");
        scanf("%d",&employees[*count].emp_id);
        printf("Enter name of the employee: ");
        getchar();
        fgets(employees[*count].name,sizeof(employees[*count].name),stdin);
        employees[*count].name[strcspn(employees[*count].name,"\n")] = 0;
        printf("Enter the salary: ");
        scanf("%f",&employees[*count].salary);
        *count++;
        printf("Employee added successfully!\n");

    }
}
```

```
void updateSalary(struct Employee employees[],float salary,int id,int count){
    if(count==0){
        printf("No records available!");
    }
    for(int i=0;i<count;i++){
        if(employees[i].emp_id==id){
            employees[i].salary=salary;
            printf("Updated salary: %.2f",employees[i].salary);
            printf("Salary updated successfully!");
        }
        else{
            printf("No such employee");
        }
    }
}
void display(struct Employee employees[],int count){
    if(count==0){
        printf("No records available!");
    }
    for(int i=0;i<count;i++){
        printf("Employee id: %d Name: %s Salary:
%.2f\n",employees[i].emp_id,employees[i].name,employees[i].salary);
    }


}
void find(struct Employee employees[],int count){
    if (count == 0) {
        printf("No employees to display.\n");
    }else {
        int maxIndex = 0;
        for (int i = 1; i < count; i++) {
```

```
        if (employees[i].salary > employees[maxIndex].salary) {
            maxIndex = i;
        }
    }
    printf("Employee with the highest salary:\n");
    printf("ID: %d, Name: %s, Salary: %.2f\n", employees[maxIndex].emp_id,
employees[maxIndex].name, employees[maxIndex].salary);
  }
}
```

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.

```
 #include <stdio.h>
#include <string.h>
struct Book {
  int book_id;
  char title[100];
  char author[50];
  int copies;
```

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

int main() {
    struct Book books[50];
    int count = 0, choice;
    char cont;

    do {
        printf("\nLibrary System Menu:\n");
        printf("1. Add a book\n");
        printf("2. Issue a book\n");
        printf("3. Return a book\n");
        printf("4. Search for a book\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addBook(books, &count);
                break;
            case 2:
                issueBook(books, count);
                break;
            case 3:
                returnBook(books, count);
                break;
            case 4:
```

```
            searchBook(books, count);
            break;
        default:
            printf("Invalid option!\n");
    }
    printf("\nDo you want to continue? (y/n): ");
    getchar();
    scanf("%c", &cont);

  } while (cont == 'y' || cont == 'Y');

  printf("Goodbye!\n");
  return 0;
}
void addBook(struct Book books[], int *count) {
  if (*count >= 50) {
    printf("Library is full, cannot add more books.\n");
  } else {
    printf("Enter book ID: ");
    scanf("%d", &books[*count].book_id);
    printf("Enter book title: ");
    getchar();
    fgets(books[*count].title, sizeof(books[*count].title), stdin);
    books[*count].title[strcspn(books[*count].title, "\n")] = 0;
    printf("Enter author name: ");
    fgets(books[*count].author, sizeof(books[*count].author), stdin);
    books[*count].author[strcspn(books[*count].author, "\n")] = 0;
    printf("Enter number of available copies: ");
    scanf("%d", &books[*count].copies);
    (*count)++;
    printf("Book added successfully!\n");
```

```
    }
}
void issueBook(struct Book books[], int count) {
    int book_id, found = 0;
    printf("Enter book ID to issue: ");
    scanf("%d", &book_id);

    for (int i = 0; i < count; i++) {
        if (books[i].book_id == book_id) {
            found = 1;
            if (books[i].copies > 0) {
                books[i].copies--;
                printf("Book issued successfully! Remaining copies: %d\n", books[i].copies);
            } else {
                printf("No copies available for this book.\n");
            }
            break;
        }
    }

    if (!found) {
        printf("No book found with ID %d.\n", book_id);
    }
}
void returnBook(struct Book books[], int count) {
    int book_id, found = 0;
    printf("Enter book ID to return: ");
    scanf("%d", &book_id);

    for (int i = 0; i < count; i++) {
        if (books[i].book_id == book_id) {
```

```
            found = 1;
            books[i].copies++;
            printf("Book returned successfully! Available copies: %d\n", books[i].copies);
            break;
        }
    }

    if (!found) {
        printf("No book found with ID %d.\n", book_id);
    }
}
void searchBook(struct Book books[], int count) {
    int choice;
    char search_term[100];

    printf("Search by:\n1. Title\n2. Author\nEnter your choice: ");
    scanf("%d", &choice);
    getchar();

    if (choice == 1) {
        printf("Enter book title to search: ");
        fgets(search_term, sizeof(search_term), stdin);
        search_term[strcspn(search_term, "\n")] = 0;
        int found = 0;
        for (int i = 0; i < count; i++) {
            if (strstr(books[i].title, search_term) != NULL) {
                printf("Book found: ID: %d, Title: %s, Author: %s, Copies: %d\n",
                    books[i].book_id, books[i].title, books[i].author, books[i].copies);
                found = 1;
            }
        }
```

```
        if (!found) {
            printf("No books found with the title \"%s\".\n", search_term);
        }
    } else if (choice == 2) {
        printf("Enter author name to search: ");
        fgets(search_term, sizeof(search_term), stdin);
        search_term[strcspn(search_term, "\n")] = 0;
        int found = 0;
        for (int i = 0; i < count; i++) {
            if (strstr(books[i].author, search_term) != NULL) {
                printf("Book found: ID: %d, Title: %s, Author: %s, Copies: %d\n",
                    books[i].book_id, books[i].title, books[i].author, books[i].copies);
                found = 1;
            }
        }
        if (!found) {
            printf("No books found by the author \"%s\".\n", search_term);
        }
    } else {
        printf("Invalid choice!\n");
    }
}
```

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 <string.h>
struct Player {
    char name[50];
    int matches;
    int runs;
    float average;
};
void inputPlayerDetails(struct Player players[], int n);
void calculateAverage(struct Player players[], int n);
void displayPlayerDetails(struct Player players[], int n);
void findHighestAveragePlayer(struct Player players[], int n);
int main() {
    int n;
    char cont;
    do {

        printf("Enter the number of players: ");
        scanf("%d", &n);

        struct Player players[n];
        inputPlayerDetails(players, n);
        calculateAverage(players, n);
        displayPlayerDetails(players, n);
        findHighestAveragePlayer(players, n);
        printf("\nDo you want to continue? (y/n): ");
```

```
        getchar();
        scanf("%c", &cont);

    } while (cont == 'y' || cont == 'Y');

    printf("Goodbye!\n");
    return 0;
}
void inputPlayerDetails(struct Player players[], int n) {
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for player %d:\n", i + 1);
        printf("Name: ");
        getchar();
        fgets(players[i].name, sizeof(players[i].name), stdin);
        players[i].name[strcspn(players[i].name, "\n")] = 0;

        printf("Matches played: ");
        scanf("%d", &players[i].matches);

        printf("Total runs scored: ");
        scanf("%d", &players[i].runs);
    }
}
void calculateAverage(struct Player players[], int 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;
        }
    }
```

```
}
void displayPlayerDetails(struct Player players[], int n) {
    printf("\nPlayer Details:\n");
    for (int i = 0; i < n; i++) {
        printf("Player %d: %s\n", i + 1, players[i].name);
        printf("Matches Played: %d\n", players[i].matches);
        printf("Total Runs: %d\n", players[i].runs);
        printf("Batting Average: %.2f\n\n", players[i].average);
    }
}
void findHighestAveragePlayer(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("Player with the highest batting average:\n");
    printf("Name: %s\n", players[highestIndex].name);
    printf("Matches Played: %d\n", players[highestIndex].matches);
    printf("Total Runs: %d\n", players[highestIndex].runs);
    printf("Batting Average: %.2f\n", players[highestIndex].average);
}
```

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 <string.h>
struct Student {
   int roll_no;
   char name[50];
   float marks[5];
   char grade;
};
void inputStudentDetails(struct Student students[], int *n);
void calculateGrade(struct Student students[], int n);
void displayStudentDetails(struct Student students[], int n);
void findHighestGrade(struct Student students[], int n);

int main() {
   struct Student students[100];
   int n = 0;
   int choice;
   char cont;

   do {
      printf("\n--- Student Management System ---\n");
      printf("1. Input student details\n");
      printf("2. Calculate grades\n");
      printf("3. Display all student details\n");
      printf("4. Find student with highest grade\n");
```

```
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch(choice) {
   case 1:
      inputStudentDetails(students, &n);
      break;
   case 2:
      if (n > 0) {
         calculateGrade(students, n);
         printf("Grades calculated successfully!\n");
      } else {
         printf("No student details available. Please input student data first.\n");
      }
      break;
   case 3:
      if (n > 0) {
         displayStudentDetails(students, n);
      } else {
         printf("No student details available. Please input student data first.\n");
      }
      break;
   case 4:
      if (n > 0) {
         findHighestGrade(students, n);
      } else {
         printf("No student details available. Please input student data first.\n");
      }
      break;
   case 5:
```

```
            printf("Exiting the program. Goodbye!\n");
            return 0;
        default:
            printf("Invalid choice! Please try again.\n");
    }
    printf("\nDo you want to continue? (y/n): ");
    getchar();
    scanf("%c", &cont);

  } while (cont == 'y' || cont == 'Y');

  return 0;
}
void inputStudentDetails(struct Student students[], int *n) {
  int num;
  printf("\nEnter the number of students: ");
  scanf("%d", &num);

  for (int i = *n; i < *n + num; i++) {
    printf("\nEnter details for student %d:\n", i + 1);
    printf("Roll Number: ");
    scanf("%d", &students[i].roll_no);

    printf("Name: ");
    getchar();
    fgets(students[i].name, sizeof(students[i].name), stdin);
    students[i].name[strcspn(students[i].name, "\n")] = 0;

    printf("Enter marks for 5 subjects:\n");
    for (int j = 0; j < 5; j++) {
      printf("Subject %d: ", j + 1);
```

```
            scanf("%f", &students[i].marks[j]);
        }
    }

    *n += num;
}
void calculateGrade(struct Student students[], int n) {
    for (int i = 0; i < n; i++) {
        float total = 0;
        for (int j = 0; j < 5; j++) {
            total += students[i].marks[j];
        }
        float average = total / 5.0;
        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 displayStudentDetails(struct Student students[], int n) {
    printf("\nStudent Details:\n");
    for (int i = 0; i < n; i++) {
        printf("\nStudent %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);
    }
}
void findHighestGrade(struct Student students[], int n) {
    if (n == 0) {
        printf("No students available.\n");
        return;
    }

    int highestIndex = 0;

    for (int i = 1; i < n; i++) {
        if (students[i].grade < students[highestIndex].grade) {
            highestIndex = i;
        }
    }

    printf("\nStudent with the highest grade:\n");
    printf("Roll Number: %d\n", students[highestIndex].roll_no);
    printf("Name: %s\n", students[highestIndex].name);
    printf("Marks: ");
    for (int i = 0; i < 5; i++) {
        printf("%.2f ", students[highestIndex].marks[i]);
    }
    printf("\nGrade: %c\n", students[highestIndex].grade);
}
```

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.

```
#include <stdio.h>
#include <string.h>
struct Flight {
   char flight_number[10];
   char destination[50];
   int available_seats;
};
void addFlight(struct Flight flights[], int *n);
void bookTicket(struct Flight flights[], int n);
void cancelTicket(struct Flight flights[], int n);
void displayFlightByDestination(struct Flight flights[], int n);

int main() {
   struct Flight flights[100];
```

```
int n = 0;
int choice;
char cont;

do {
   printf("\n--- Flight Reservation System ---\n");
   printf("1. Add a new flight\n");
   printf("2. Book a ticket\n");
   printf("3. Cancel a ticket\n");
   printf("4. Display flight details by destination\n");
   printf("5. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch(choice) {
      case 1:
         addFlight(flights, &n);
         break;
      case 2:
         if (n > 0) {
            bookTicket(flights, n);
         } else {
            printf("No flights available. Please add flights first.\n");
         }
         break;
      case 3:
         if (n > 0) {
            cancelTicket(flights, n);
         } else {
            printf("No flights available. Please add flights first.\n");
         }
```

```
            break;
        case 4:
            if (n > 0) {
                displayFlightByDestination(flights, n);
            } else {
                printf("No flights available. Please add flights first.\n");
            }
            break;
        case 5:
            printf("Exiting the program. Goodbye!\n");
            return 0;
        default:
            printf("Invalid choice! Please try again.\n");
    }
    printf("\nDo you want to continue? (y/n): ");
    getchar();
    scanf("%c", &cont);

  } while (cont == 'y' || cont == 'Y');

  return 0;
}
void addFlight(struct Flight flights[], int *n) {
  printf("\nEnter flight number: ");
  getchar();
  fgets(flights[*n].flight_number, sizeof(flights[*n].flight_number), stdin);
  flights[*n].flight_number[strcspn(flights[*n].flight_number, "\n")] = 0;

  printf("Enter destination city: ");
  fgets(flights[*n].destination, sizeof(flights[*n].destination), stdin);
  flights[*n].destination[strcspn(flights[*n].destination, "\n")] = 0;
```

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

    (*n)++;
    printf("Flight added successfully!\n");
}
void bookTicket(struct Flight flights[], int n) {
    char flight_number[10];
    int found = 0;

    printf("\nEnter flight number to book ticket: ");
    getchar();
    fgets(flight_number, sizeof(flight_number), stdin);
    flight_number[strcspn(flight_number, "\n")] = 0;

    for (int i = 0; i < n; i++) {
        if (strcmp(flights[i].flight_number, flight_number) == 0) {
            found = 1;
            if (flights[i].available_seats > 0) {
                flights[i].available_seats--;
                printf("Ticket booked successfully!\n");
                printf("Remaining seats: %d\n", flights[i].available_seats);
            } else {
                printf("No available seats on this flight.\n");
            }
            break;
        }
    }

    if (!found) {
```

```
        printf("Flight with number %s not found.\n", flight_number);
    }
}
void cancelTicket(struct Flight flights[], int n) {
    char flight_number[10];
    int found = 0;

    printf("\nEnter flight number to cancel ticket: ");
    getchar();
    fgets(flight_number, sizeof(flight_number), stdin);
    flight_number[strcspn(flight_number, "\n")] = 0;

    for (int i = 0; i < n; i++) {
        if (strcmp(flights[i].flight_number, flight_number) == 0) {
            found = 1;
            flights[i].available_seats++;
            printf("Ticket canceled successfully!\n");
            printf("Remaining seats: %d\n", flights[i].available_seats);
            break;
        }
    }

    if (!found) {
        printf("Flight with number %s not found.\n", flight_number);
    }
}
void displayFlightByDestination(struct Flight flights[], int n) {
    char destination[50];
    int found = 0;

    printf("\nEnter destination city: ");
```

```
    getchar();
    fgets(destination, sizeof(destination), stdin);
    destination[strcspn(destination, "\n")] = 0;

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

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