DAY 12

1)**calloc**

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int *p=NULL;
  int n;

  printf("enter the number of integers that want to
store:");
  scanf("%d",&n);

  p=(int*)calloc(n,sizeof(int));

  for(int i=0;i<n;i++){
      printf("p[%d]=%d\t",i,*(p+i));
  }


    return 0;
}
```

2)REALLOC

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
  char *str;

  str=(char*)malloc(15*sizeof(char));

  strcpy(str,"jason");

  printf("string=%s,address=%p\n",*str,str);


   str=(char*)realloc(str,25);

   strcat(str,".com");

   printf("string=%s,address=%p",str,str);


  free(str);

    return 0;
}
```

3)double pointer

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
  int i=4,j=6,k=7;
  int **ipp=NULL;
  int *ip1=&i;
  int *ip2=&j;

  ipp=&ip1;


  printf("value pointed by ip1:%d\n",*ip1);

  printf("value pointed by ip2:%d\n",*ip2);

  printf("value pointed by ipp:%d\n",**ipp);

  **ipp=8;

  printf("value pointed by ipp:%d",**ipp);


    return 0;
}
```

4)tripple pointer

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
  int i=4,j=6,k=7;
  int **ipp=NULL;
  int ***ipp1=NULL;
  int *ip1=&i;
  int *ip2=&j;

  ipp=&ip1;

  ipp1=&ipp;

  printf("value pointed by ip1:%d\n",*ip1);

  printf("value pointed by ip2:%d\n",*ip2);

  printf("value pointed by ipp:%d\n",**ipp);

  **ipp=8;

  printf("value pointed by ipp:%d\n",**ipp);

  ***ipp1=9;

  printf("value pointed by ipp:%d",***ipp1);
```

```
    return 0;
}
```

ASSIGNMENT

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

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int n;
    char c;
    printf("enter the size of the array:");
    scanf("%d",&n);
```

```
    int *ptr=(int*)malloc(n*sizeof(int));
    printf("Enter %d elements for the array:\n", n);
    for(int i=0;i<n;i++){
                scanf("%d",&ptr[i]);
            }
            for(int i=0;i<n;i++){
                printf("a[%d]=%d",i,*(ptr+i));
            }
    printf("\n");
    printf("choose option:");
    scanf(" %c",&c);
    switch(c){
        case 'i':
            int newsize;
            printf("Enter the new size to increase the
array: \n");
            scanf("%d",&newsize);
            ptr=(int*)realloc(ptr,newsize*(sizeof(int)));

            printf("Enter %d elements for the array:\n",
newsize);
            for(int i=0;i<newsize;i++){
                scanf("%d",&ptr[i]);
            }
            for(int i=0;i<newsize;i++){
                printf("a[%d]=%d",i,*(ptr+i));
            }
            break;
        case 'd':
```

```
            printf("Enter the new size to increase the
array: \n");
            scanf("%d",&newsize);
            ptr=(int*)realloc(ptr,newsize*(sizeof(int)));
            printf("Enter %d elements for the array:\n", n);
            for(int i=0;i<newsize;i++){
                scanf("%d",&ptr[i]);
            }
            for(int i=0;i<newsize;i++){
                printf("a[%d]=%d",i,*(ptr+i));
            }
            break;

    return 0;
}
}
```

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.

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

int main() {
    char *str1, *str2;


    printf("Enter the first string: ");
    str1 = (char *)malloc(15 * sizeof(char));

    fgets(str1, 15, stdin);



    printf("Enter the second string: ");
    str2 = (char *)malloc(15 * sizeof(char));

    fgets(str2, 15, stdin);

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


    strcat(str2, str1);
```

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


    free(str1);
    free(str2);

    return 0;
}
```

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 = (char)malloc(100 * sizeof(char));
 if (str1 == NULL) {
 printf("Memory allocation failed\n");
```

```
return 1;
}
char str2[100];
printf("Enter the first string: ");
scanf("%[^\n]", str1);
getchar();
printf("Enter the second string: ");
scanf("%[^\n]", str2);
str1 = (char*)realloc(str1,(strlen(str2)+1));
strcat(str1, str2);
printf("Concatenated string: %s\n", str1);
free(str1);
return 0;
}
```

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);
 }
```

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:
   - ○ Addanodetothe 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 rows, cols;
```

```
int **array;
 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);
 for (int j = 0; j < i; j++) {
 free(array[j]);
 }
 free(array);
 return 1;
 }
 }
 printf("Enter values for the 2D array:\n");
 for (int i = 0; i < rows; i++) {
 for (int j = 0; j < cols; j++) {
 printf("Enter value for array[%d][%d]: ", i, j);
 scanf("%d", &array[i][j]);
 }
 }
 printf("The 2D array 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;
 }
```

Structure

1)

```
#include<stdio.h>
struct date{
  int day;
  int month;
  int year;
};
int main()
```

```
{
   struct date today;
   printf("size of today=%ld",sizeof(today));



}
```

2)

```
#include<stdio.h>
struct date{
  int day;
  int month;
  int year;
};
int main()
{
   struct date today;

   today.day=21;
   today.month=11;
   today.year=2024;

   printf("size of today=%ld\n",sizeof(today));
```

```
   printf("todays
date=%d-%d-%d",today.day,today.month,toda
y.year);

}
```

6)

```
#include<stdio.h>
//defining structure
struct student {
   char name[50];
   int rollno;
   float marks;
};
int main(){

   //intialisation of struct
    struct student s1[50];
    int choice;
    int count=0;

    while(1){
        printf("1.Add student\n");
```

```
        printf("2.Display all student\n");
        printf("3.find student by roll number\n");
        printf("4.calculate average marks\n");
        printf("5.Exit\n");
        printf("enter option:");
        scanf("%d",&choice);
        switch(choice){
            case 1:
              printf("enter name:");
              scanf(" %[^\n]", s1[count].name);


              printf("enter roll no:");
              scanf("%d",&s1[count].rollno);

              printf("enter marks:");
              scanf("%f",&s1[count].marks);

              count++;
              printf("student added
successfully\n");
              printf("\n");
              break;
```

```
            case 2:
                for(int i=0;i<count;i++){
                    printf("name:%s\troll
no:%d\tmarks:%f\n",s1[i].name,s1[i].rollno,s1[i
].marks);
                }
                printf("\n");
                break;

            case 3:
                int roll_check;
                printf("enter roll no to check");
                scanf("%d",&roll_check);
                for(int i=0;i<count;i++){
                  if(s1[i].rollno==roll_check){
                      printf("hi %s\n",s1[i].name);
                   }
                   else{
                      printf("enter valid roll no");
                   }
                }
                printf("\n");
                break;
            case 4:
```

```
                float average;
                float sum=0;
                for(int i=0;i<count;i++){
                    sum=sum+s1[i].marks;
                }
                average=sum/count;
                printf("average
marks:%.2f\n",average);
                printf("\n");
                break;

            case 5:
                printf("EXITING FROM THE
SYSTEM");
                printf("\n");
        }









    }
```

```
    return 0;
}
```

7)

```
int main(){

    //struct date today;

    struct student s1 = {.rollNumber = 1234,
.name = "Abhinav", .marks = 95.5};

    printf("S1's Name roll number and marks is
%s %d & %f \n", s1.name,
s1.rollNumber,s1.marks);

    return 0;

}
```

```
9)#include<stdio.h>

struct coordinate{
  int x;
  int y;

};

void print_coordinate(struct coordinate);

int main(){

    //compound literal

    print_coordinate((struct coordinate){5,6});

    /*struct coordinate pointA={5,6};

    print_coordinate(pointA);

    return 0;*/
}
void print_coordinate(struct coordinate temp){
```

```
    printf("x=%d\ty=%d",temp.x,temp.y);

}
10)#include<stdio.h>

struct coordinate{//declaration of a structure
  int x;
  int y;

};



int main(){

 struct coordinate pnt[5];//intialise structure

 for(int i=0;i<5;i++){
    printf("intialise the struct present in the %d
index\n",i);
    scanf("%d%d",&pnt[i].x,&pnt[i].y);
 }

 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);

 }

}
```

11)ARRAY OF STRUCTURES

```
#include<stdio.h>

struct coordinate{//declaration of a structure
  int x;
  int y;

};



int main(){

 //struct coordinate
pnt[5]={{12,10},{13,14},{15,16},{19,20},{21,22}
};
```

```
 struct coordinate
pnt[5]={[2]={12,10},[1]={14,15},[3]={17,18},[0]
={4,5}};

 for(int i=0;i<5;i++){
     printf("coordinates at %d
index:(%d,%d)\n",i,pnt[i].x,pnt[i].y);
 }

}
```

11)array in structures

```
#include<stdio.h>

struct calender{//declaration of a structure
  int days;
  char name[3];

};



int main(){
    struct calender Months[12];
    for(int i=0;i<12;i++){
```

```
        printf("enter the month name and no of
days:");
        scanf("%s
%d",Months[i].name,&Months[i].days);
        printf("\n");
    }
    for(int i=0;i<12;i++){
        printf("month
name=%s\tDays=%d",Months[i].name,Months
[i].days);
        printf("\n");
    }

}
```

9)nested structures

```
#include<stdio.h>

struct currentDate{//declaration of a structure
  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={{21,22,2024},{51,01,17}};



    printf("current
date=%d-%d-%d",dt.d1.day,dt.d1.month,dt.d1
.year);
```

```
    printf("current
time=%d-%d-%d",dt.t1.sec,dt.t1.min,dt.t1.hour
s);









}
```

12)NESTING STRUCTURE

```
#include<stdio.h>


struct currentTime{
   struct currentDate{//declaration of a
structure
       int day;
        int month;
        int year;
   };
```

```
    int sec;
   int min;
   int hours;
};
struct CDateTime{
    struct currentDate d1;
    struct currentTime t1;
};


int main(){

    struct CDateTime
dt={{21,22,2024},{51,01,17}};



    printf("current
date=%d-%d-%d",dt.d1.day,dt.d1.month,dt.d1
.year);

     printf("current
time=%d-%d-%d",dt.t1.sec,dt.t1.min,dt.t1.hour
s);
```

}

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

};
int main(){
    struct employee e1[60];
    int count=0,choice;
    while(1){
        printf("1.Add Employee\n");
        printf("2.Update Employee Salary by ID\n");
        printf("3.Display Employee Details\n");
        printf("4.Highest Salary\n");
        printf("5.Exit\n");
        printf("enter your choice:");
        scanf("%d",&choice);
        switch(choice){
            case 1:
```

```
                printf("enter employee id:");
                scanf("%d",&e1[count].id);

                getchar();

                printf("enter employee name:");
                scanf("%[^\n]",e1[count].name);

                printf("Enter salary:");
                scanf("%f",&e1[count].salary);

                count++;

                printf("Employee Added
Successfully");




                printf("\n");
                break;
```

```
            case 2:
                int employee_id;
                float new_salary;
                printf("enter employee id to update
the value:");
                scanf("%d",&employee_id);
                for(int i=0;i<count;i++){
                    if(employee_id==e1[i].id){
                       printf("enter new salary:");
                       scanf("%f",&new_salary);
                       printf("salary updated");
                     }
                    else{
                        printf("employee id not
found");
                    }
                }
                printf("\n");
                break;

            case 3:
                for(int i=0;i<count;i++){
```

```
printf("id=%d\temployee-name=%s\tsalary=%.
2f\n",e1[i].id,e1[i].name,e1[i].salary);
                }
                printf("\n");
                break;

            case 4:
                float max=e1[0].salary;
                for(int i=0;i<count;i++){
                    if(e1[i].salary>max){
                        max=e1[i].salary;
                    }
                }
                printf("highest salary=%.2f",max);
                printf("\n");
                break;

            case 5:
                printf("exiting from the system");
                break;
        }
    }
```

```
}

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

struct library {
    int id;
    char title[100];
    char name[50];
    int copies;
};

int main() {
    struct library b1[60];
    int count = 0, choice;
    char input[100];

    while (1) {
        printf("\n1. Add Book\n");
        printf("2. Issue a book by reducing no of
copies\n");
        printf("3. Return a book by increasing no
of copies\n");
```

```
        printf("4. Search a book by title or author
name\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();  // Clear newline character
from input buffer

        switch (choice) {
            case 1:
                printf("Enter book ID: ");
                scanf("%d", &b1[count].id);
                getchar();  // Clear newline
character

                printf("Enter book title: ");
                scanf("%[^\n]", b1[count].title);
                getchar();

                printf("Enter author name: ");
                scanf("%[^\n]", b1[count].name);
                getchar();

                printf("Enter number of copies: ");
```

```
            scanf("%d", &b1[count].copies);

            count++;
            printf("Book added successfully!\n");
            break;

        case 2:
            printf("Enter the name of the book
to issue: ");
            scanf("%[^\n]", input);
            getchar();

            int issued = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(input, b1[i].title) == 0) {
                    if (b1[i].copies > 0) {
                        b1[i].copies--;
                        printf("Book '%s' issued
successfully!\n", b1[i].title);
                    } else {
                        printf("Book '%s' is out of
stock!\n", b1[i].title);
                    }
                    issued = 1;
```

```
                    break;
                }
            }
            if (!issued) {
                printf("Book '%s' not found!\n",
input);
            }
            break;

        case 3:
            printf("Enter the name of the book
to return: ");
            scanf("%[^\n]", input);
            getchar();

            int returned = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(input, b1[i].title) == 0) {
                    b1[i].copies++;
                    printf("Book '%s' returned
successfully!\n", b1[i].title);
                    returned = 1;
                    break;
                }
```

```
            }
            if (!returned) {
                printf("Book '%s' not found in the
library!\n", input);
            }
            break;

        case 4:
            printf("Enter title or author name to
search: ");
            scanf("%[^\n]", input);
            getchar();

            int found = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(input, b1[i].title) == 0 ||
strcmp(input, b1[i].name) == 0) {
                    printf("\nBook Found:\n");
                    printf("ID: %d\nTitle:
%s\nAuthor: %s\nCopies: %d\n",
                           b1[i].id, b1[i].title,
b1[i].name, b1[i].copies);
                    found = 1;
                }
```

```
            }
            if (!found) {
                printf("No book found with title or
author '%s'.\n", input);
            }
            break;

        case 5:
            printf("Exiting from the system.
Goodbye!\n");
            return 0;

        default:
            printf("Invalid choice! Please try
again.\n");
        }
    }
}
```

5)/*Problem 5: Flight Reservation System
Objective: Simulate a simple flight reservation system using structures.
Description:
Define a structure Flight with fields:

char flight_number[10]: Flight number
char destination[50]: Destination city
int available_seats: Number of available seats
Write a program to:
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.
has context menu*/

```
#include<stdio.h>
#include<string.h>
struct flight{
  char flight_number[10];
  char destination[50];
  int seats;

};
int main(){
   struct flight f1[20];
   int choice;
   int count=0;
   while(1){
```

```
        printf("1.Add flights\n");
        printf("2.book flights\n");
        printf("3.display flight details\n");
        printf("4.cancel tickets\n");
        printf("5.exit\n");
        printf("enter choice:");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                printf("Flight number: ");
                getchar();
                scanf("%[^\n]",
f1[count].flight_number);

                printf("Enter your destination: ");
                getchar();
                scanf("%[^\n]",
f1[count].destination);

                printf("Enter number of seats: ");
                scanf("%d", &f1[count].seats);

                count++;
```

```
            printf("Flight added
successfully!\n");
            printf("\n");
            break;

        case 2:
             char flight[30];
            printf("Enter your flight number: ");
            getchar();
            scanf("%[^\n]", flight);

            //int booked = 0;

            for (int i = 0; i < count; i++) {
                if (strcmp(flight,
f1[i].flight_number) == 0) {

                    f1[i].seats--;
                    printf("Flight booked
successfully! Remaining seats: %d\n",
f1[i].seats);
                } else {
                    printf("No seats available on
flight %s.\n", f1[i].flight_number);
```

```
                }

            }
             break;
             printf("\n");




        case 3:
            char dest[40];
            printf("enter destination:");
            getchar();
            scanf("%[^\n]",dest);

            for(int i=0;i<count;i++){

if(strcmp(dest,f1[i].destination)==0){
                printf("Flight Number:%s\t
seats:%d",f1[i].flight_number,f1[i].seats);
            }else{
                printf("enter valid
destination\n");
            }
```

```
            }
            printf("\n");
            break;

        case 4:

            char flight1[30];
            printf("Enter your flight number: ");
            getchar();
            scanf("%[^\n]", flight1);



            for (int i = 0; i < count; i++) {
                if (strcmp(flight1,
f1[i].flight_number) == 0) {

                    f1[i].seats++;
                    printf("Flight cancelled
successfully! Remaining seats: %d\n",
f1[i].seats);
                } else {
```

```
                        printf("No seats available on
flight %s.\n", f1[i].flight_number);
                    }

                }
                 break;
                 printf("\n");

            case 5:

                printf("EXIT\n");
        }




    }
```

```
    return 0;
}
```

4)Problem 3: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. Define a structure Player with fields:

o char name[50]: Player name

o int matches: Number of matches played

o int runs: Total runs scored

o float average: Batting average

2. Write a program to:

o Input details for n players.

o Calculate and display the batting average for each player.

o Find and display the player with the highest batting average.

```
#include <stdio.h>
struct Player {
char name[50];
int matches;
int runs;
float average;
};
```

```
int main() {
int n;
printf("Enter the number of players: ");
scanf("%d", &n);
struct Player players[n];
int highestIndex = 0;
for (int i = 0; i < n; i++) {
printf("Enter details for player %d\n", i + 1);
printf("Name: ");
scanf(" %[^\n]", players[i].name);
printf("Matches played: ");
scanf("%d", &players[i].matches);
printf("Total runs: ");
scanf("%d", &players[i].runs);
if (players[i].matches > 0)
players[i].average = (float)players[i].runs /
players[i].matches;
else
players[i].average = 0;
if (players[i].average >
players[highestIndex].average) {
highestIndex = i;
}
}
```

```
printf("\nPlayer Statistics:\n");
for (int i = 0; i < n; i++) {
printf("Name: %s, Matches: %d, Runs: %d,
Average: %.2f\n",
players[i].name, players[i].matches,
players[i].runs, players[i].average);
}
printf("\nPlayer with the highest batting
average:\n");
printf("Name: %s, Average: %.2f\n",
players[highestIndex].name,
players[highestIndex].average);
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:
   - o int roll_no: Roll number
   - o char name[50]: Student name
   - o float marks[5]: Marks in 5 subjects
   - o char grade: Grade based on the average marks

2. Write a program to:
o Input details of n students.
o Calculate the average marks and assign grades (A, B, C, etc.).
o Display details of students along with their grades.

```
#include <stdio.h>
struct Student {
int roll_no;
char name[50];
float marks[5];
char grade;
};
int main() {
int n;
printf("Enter the number of students: ");
scanf("%d", &n);
struct Student students[n];
for (int i = 0; i < n; i++) {
printf("Enter roll number and name for student %d: ", i + 1);
scanf("%d %[^\n]", &students[i].roll_no, students[i].name);
float total = 0;
```

```
printf("Enter marks for 5 subjects: ");
for (int j = 0; j < 5; j++) {
scanf("%f", &students[i].marks[j]);
total += students[i].marks[j];
}
float average = total / 5;
if (average >= 90) students[i].grade = 'A';
else if (average >= 75) students[i].grade = 'B';
else if (average >= 50) students[i].grade = 'C';
else students[i].grade = 'D';
}
printf("\nStudent Details:\n");
for (int i = 0; i < n; i++) {
printf("Roll No: %d, Name: %s, Grade: %c\n",
students[i].roll_no, students[i].name,
students[i].grade);
}
return 0;
}
```

Problem 5: F