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 *ptr=NULL;
   int n,i,n1,op;


   printf("enter the number of ekements in the aray:\n");
   scanf("%d",&n);

   ptr=(int *)malloc(n*sizeof(int));
   if(ptr==NULL){
      printf("memory is not allocated\n");
   }else{
      printf("memory allocated successfully\n");
   }
   printf("enter the array elements\n");
   for(i=0;i<n;i++){
      scanf("%d",&ptr[i]);
   }
  printf("array elements\n[");
```

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

printf("Do you want to increse the size of elements 1.yes/2.no ?\n");
scanf("%d",&op);
switch (op)
{
case 1:
   printf("enter the elements you want to add\n");
   scanf("%d",&n1);

   ptr=(int*)realloc(ptr,n1);
   printf("enter the new wlwmwnts");
   for(i=0;i<n1;i++){
   scanf("%d",&ptr[i]);
    }
     printf("array elements\n[");
     for(i=0;i<n1;i++){
     printf("%d\t",ptr[i]);
     }printf("]");

   break;
case 2:
  exit(0);
  break;


default:
printf("invalid option");
   break;}}
```

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>
void my_strcat(char *, char*);
int main()
{
char *str1, *str2;
str1 = (char *)malloc(10 * sizeof(char));
str2 = (char *)malloc(10 * sizeof(char));
printf("Enter the 1st string: ");
scanf("%[^\n]", str1);
getchar();
printf("Enter the 2nd string: ");
scanf("%[^\n]", str2);
my_strcat(str1, str2);
printf("The concatenated string is : %s ",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 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 r, c;
printf("Enter the number of row: ");
scanf("%d", &r);
printf("Enter the number of column: ");
scanf("%d", &c);
int **arr=(int **)malloc(r * sizeof(int *));
for(int i=0; i<r; i++)
{
arr[i]=malloc(c* sizeof(int));
}
printf("Enter the array elements: \n");
for(int i=0; i<r; i++)
```

```
{
for(int j=0; j<c; j++)
{
scanf("%d", &arr[i][j]);
}
}
printf("The array elements are: \n");
for(int i=0; i<r; i++)
{
for(int j=0; j<c; j++)
{
printf("%d\t", arr[i][j]);
}
printf("\n");
}
for(int i=0; i<r; i++)
{
free(arr[i]);
}
free(arr);
}
```

Problem : 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!

```
#include <stdio.h>
#include <stdlib.h>
void add_student();
void display_student();
void find_student();
void find_averageMark();
```

```
struct student
{
   char name[50];
   int rollno;
   float mark;
} students[50];

int count = 0;

int main()
{
   int option;

   while (1)
   {
      printf("1. Add Student\n2. Display All Students\n3. Find Student by Roll Number\n4. Calculate Average Marks\n5. Exit\n");
      scanf("%d", &option);

      switch (option)
      {
      case 1:
         add_student();
         break;
      case 2:
         display_student();
         break;
      case 3:
         find_student();
         break;
      case 4:
```

```
            find_averageMark();
            break;
        case 5:
            exit(0);
            break;

        default:
            printf("Invalid option.Check again\n");
            printf("\n");
            break;
        }
    }
}

void add_student()
{
    struct student newStudent;
    printf("ADD STUDENT\n");
    printf("Enter student name: ");
    scanf("%s", newStudent.name);
    printf("Enter roll number: ");
    scanf("%d", &newStudent.rollno);
    printf("Enter score: ");
    scanf("%f", &newStudent.mark);

    students[count++] = newStudent;
    printf("Student added succesfully\n");
    printf("\n");
}

void display_student()
```

```
{
    if (count == 0)
    {
        printf("No student is there in the list");
    }
    else
    {
        printf("STUDENT DETAILS\n");
        printf("RollNmber\tName\tMarks\n");
        for (int i = 0; i < count; i++)
        {
            printf("%d\t\t%s\t%.2f\n", students[i].rollno, students[i].name, students[i].mark);
        }
    }
    printf("\n");
}

void find_student()
{
    int rol;
    printf("Enter the roll number:");
    scanf("%d", &rol);

    for (int i = 0; i < count; i++)
    {
        if (rol == students[i].rollno)
        {
            printf("Name : %s\n", students[i].name);
            printf("Mark : %.2f\n", students[i].mark);
        }
    }
```

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

void find_averageMark()
{
    int mark = 0;
    for (int i = 0; i < count; i++)
    {
        mark = mark + students[i].mark;
    }
    float average = mark / count;
    printf("Average of marks = %.2f\n", average);
    printf("\n");
}
```

Problem : 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>
#include <stdlib.h>
void add_employee();
void display_employee();
void update_salary();
void highSalary_Employee();

struct Emp
{
   int emp_id;
   char emp_name[50];
   float emp_salary;
};

struct Emp employee[50];
int count = 0;

int main()
{

   int option;

   while (1)
   {
      printf("Enter the option\n1.Add Employee\n2.Display All Employees\n3.Update Salary\n4.View Employee with High Salary\n5.Exit\n");
      scanf("%d", &option);

      switch (option)
      {
      case 1:
```

```
            add_employee();
            printf("\n");
            break;
        case 2:
            display_employee();
            printf("\n");
            break;
        case 3:
            update_salary();
            printf("\n");
            break;
        case 4:
            highSalary_Employee();
            printf("\n");
            break;
        case 5:
            exit(0);

            break;

        default:
            printf("Invalid option.\n");
            printf("\n");
            break;
        }
    }

    return 0;
}

void add_employee()
```

```
{
    struct Emp newEmployee;

    printf("Enter Employee ID: ");
    scanf("%d", &newEmployee.emp_id);
    printf("Enter name of Employee : ");
    scanf("%s", newEmployee.emp_name);
    printf("Enter Salary: ");
    scanf("%f", &newEmployee.emp_salary);

    employee[count] = newEmployee;
    count++;

    printf("Employee Successfully Added!!!!");
}

void display_employee()
{
    printf("EMPLOYEE DETAILS\n");
    printf("EMP_ID\tEMP_NAME\tEMP_SALARY\n");
    for (int i = 0; i < count; i++)
    {
        printf("%d\t%s\t%.2f\n", employee[i].emp_id, employee[i].emp_name,
employee[i].emp_salary);
    }
}

void update_salary()
{
    int id;
    float newSalary;
```

```
    printf("Enter employee Id:");
    scanf("%d", &id);

    for (int i = 0; i < count; i++)
    {
        if (id == employee[i].emp_id)
        {
            printf("Name: %s\nSalary: %.2f\n", employee[i].emp_name, employee[i].emp_salary);
            printf("Enter Updated salary: ");
            scanf("%f", &newSalary);
            employee[i].emp_salary = newSalary;
            printf("Changes added successfully");
        }
    }
}

void highSalary_Employee()
{
    float max = employee[0].emp_salary;
    char name[10];
    int id;
    for (int i = 0; i < count; i++)
    {
        if (employee[i].emp_salary > max)
        {
            max = employee[i].emp_salary;
        }
    }
    printf("Highest salary is of rupees $%.1f\n", max);
}
```

Problem : Library Management System

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

Description:

Define a structure Book with fields:

int book_id: Book ID

char title[100]: Book title

char author[50]: Author name

int copies: Number of available copies

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<stdlib.h>
#include<string.h>
void add_book();
void display_book();
void issue_book();
void return_book();
void search_book();
struct Book{
   int book_id;
   char book_name[100];
   char author[50];
   int copy;
};
int count;
struct Book Books[100];
```

```
int main(){
   int option;
   while(1){
      printf("Enter the option:\n1.Add Book\n2.Display Book\n3.Issue book\n4.Return
Book\n5.Search Book\n6.Exit\n");
      scanf("%d",&option);

      switch (option)
      {
      case 1:
         add_book();
         printf("\n");
         break;

       case 2:
         display_book();

         printf("\n");
         break;



      case 3:
         issue_book();
         printf("\n");
         break;

      case 4:
         return_book();
         printf("\n");
         break;
```

```
        case 5:
            exit(0);
            break;
        default:
        printf("invalid option");
            break;
        }
    }
}

void add_book(){
    struct Book newBook;
    printf("Enter Book ID: ");
    scanf("%d",&newBook.book_id);
    printf("Enter Book Name: ");
    scanf("%s",&newBook.book_name);
    printf("Enter Name of Author: ");
    scanf("%s",&newBook.author);
    printf("Enter number of copies");
    scanf("%d",&newBook.copy);

    Books[count]=newBook;
    count++;
    printf("Book added successfully!!!!!");
}

void display_book(){
    printf("BOOK DETAILS\n");
    printf("BOOK_ID\tBOOK_NAME\tNAME OF AUTHOR\tNO.OF COPIES\n");
    for (int i = 0; i < count; i++)
```

```
    {
        printf("%d\t%s\t\t%s\t\t%d\n", Books[i].book_id,
Books[i].book_name,Books[i].author,Books[i].copy);
    }
}

void issue_book(){
    int id;
    display_book();
    printf("Enter the ID of book:");
    scanf("%d",&id);
    for(int i=0;i<count;i++){
        if(id==Books[i].book_id){
            printf("%s book with id %d successfully issued....\n",Books[i].book_name,Books[i].book_id);
            Books[i].copy-=1;
        }else{
            printf("Enter correct book id\n");
        }
    }

}

void return_book(){
    int id;

    printf("Enter the ID of book:");
    scanf("%d",&id);
    for(int i=0;i<count;i++){
        if(id==Books[i].book_id){
            printf("%s book with id %d successfully returned....\n",Books[i].book_name,Books[i].book_id);
            Books[i].copy+=1;
```

```
    }else{
        printf("Enter correct book id\n");
    }
  }

}
```

Problem : Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

Define a structure Player with fields: char name[50]: Player name

int matches: Number of matches played

int runs: Total runs scored

float average: Batting average

Write a program to: Input details for n players.

Calculate and display the batting average for each player.

Find and display the player with the highest batting average.

```
#include <stdio.h>
#include <string.h>
struct Player {
char name[50];
int matches;
int runs;
float average;
};
int main() {
int max = 50;
struct Player players[max];
```

```
int count = 0;
int choice;
while (1) {
printf("\n1. Add player details\n");
printf("2. Display all player statistics\n");
printf("3. Find and display the player with the highest batting average\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
if (count >= max) {
printf("Cannot add more players. Maximum limit reached.\n");
} else {
printf("Enter name: ");
scanf(" %s", players[count].name);
printf("Number of matches: ");
scanf("%d", &players[count].matches);
printf("Total runs scored: ");
scanf("%d", &players[count].runs);
if (players[count].matches != 0) {
players[count].average = (float)players[count].runs / players[count].matches;
} else {
players[count].average = 0.0;
}
count++;
printf("Player details added successfully.\n");
}
break;
case 2:
if (count == 0) {
```

```
printf("No player records available.\n");
} else {
printf("\nPlayer Statistics:\n");
for (int i = 0; i < count; i++) {
printf("Name: %s, Matches: %d, Runs: %d, Average: %.2f\n", players[i].name,
players[i].matches, players[i].runs, players[i].average);
}
}
break;
case 3:
if (count == 0) {
printf("No player records available.\n");
} else {
int highestIndex = 0;
for (int i = 1; i < count; 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, Average: %.2f\n", players[highestIndex].name,
players[highestIndex].matches, players[highestIndex].runs, players[highestIndex].average);
}
break;
case 4:
return 0;
default:
printf("Invalid choice! Please try again.\n");
break;
}
} return 0; }
```

Problem 4: Student Grading System

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

Description:

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

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;
};
int main() {
int max;
printf("enter number of students \n");
scanf("%d",&max);
struct Student students[max];
int count = 0;
int choice;
int roll_no;
while (1) {
```

```
printf("\n1. input details of students\n");
printf("2. calculate average, and assign grade\n");
printf("3. Display all student details along with grades\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
if (count >= max) {
printf("Cannot add more students. Maximum limit reached.\n");
} else {
printf("Enter roll number: ");
scanf("%d", &students[count].roll_no);
printf("Enter name: ");
scanf(" %s", students[count].name);
printf("Roll number and name added successfully.\n");
count++;
}
break;
case 2:
if (count == 0) {
printf("No students available. Please add students first.\n");
} else {
printf("Enter roll number to input marks and calculate grade: ");
scanf("%d", &roll_no);
int found = 0;
for (int i = 0; i < count; i++) {
if (students[i].roll_no == roll_no) {
float total_marks = 0;
for (int j = 0; j < 5; j++) {
printf("Enter marks for subject %d: ", j + 1);
```

```
scanf("%f", &students[i].marks[j]);
total_marks += students[i].marks[j];
}
float average = total_marks / 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';
}
printf("Average marks: %.2f, Grade: %c\n", average, students[i].grade);
found = 1;
break;
}
}
if (!found) {
printf("Student with roll number %d not found.\n", roll_no);
}
}
break;
case 3:
if (count == 0) {
printf("No student records available.\n");
} else {
printf("\nStudent Details:\n");
for (int i = 0; i < count; i++) {
```

```
printf("Roll No: %d, Name: %s, Marks: [%.2f, %.2f, %.2f, %.2f, %.2f], Grade: %c\n",
students[i].roll_no, students[i].name, students[i].marks[0], students[i].marks[1],
students[i].marks[2], students[i].marksmarks[3], students[i].marks[4], students[i].grade);
}
}
break;
case 4:
return 0;
default:
printf("Invalid choice! Please try again.\n");
break;
}
}
return 0;
}
```

Problem : 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.

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

```
struct Flight {
char flight_number[10];
char destination[50];
int available_seats;
};
int main() {
int max_flights = 50;
struct Flight flights[max_flights];
int count = 0;
int choice;
while (1) {
printf("\n1. Add a flight\n");
printf("2. Book tickets for a flight\n");
printf("3. Display flight details by destination\n");
printf("4. Cancel tickets for a flight\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
if (count >= max_flights) {
printf("Cannot add more flights. Maximum limit reached.\n");
} else {
printf("Enter flight number: ");
scanf("%s", flights[count].flight_number);
printf("Enter destination: ");
scanf(" %[^\n]%*c", flights[count].destination); // Reading string with spaces
printf("Enter number of available seats: ");
scanf("%d", &flights[count].available_seats);
count++;
printf("Flight added successfully.\n");
```

```
}
break;
case 2:
case 2:
if (count == 0) {
printf("No flights available. Please add flights first.\n");
} else {
char flight_number[10];
int tickets;
printf("Enter flight number to book tickets: ");
scanf("%s", flight_number);
printf("Enter number of tickets to book: ");
scanf("%d", &tickets);
int found = 0;
for (int i = 0; i < count; i++) {
if (strcmp(flights[i].flight_number, flight_number) == 0) {
if (flights[i].available_seats >= tickets) {
flights[i].available_seats -= tickets;
printf("Tickets booked successfully. Remaining seats: %d\n", flights[i].available_seats);
} else {
printf("Not enough available seats. Remaining seats: %d\n", flights[i].available_seats);
}
found = 1;
break;
}
}
if (!found) {
printf("Flight not found.\n");
}
}
break;
```

```
case 3:
if (count == 0) {
printf("No flights available. Please add flights first.\n");
} else {
char destination[50];
printf("Enter destination to search for flights: ");
scanf(" %[^\n]%*c", destination); // Reading string with spaces
int found = 0;
for (int i = 0; i < count; i++) {
if (strcmp(flights[i].destination, destination) == 0) {
printf("Flight Number: %s, Destination: %s, Available Seats: %d\n", flights[i].flight_number,
flights[i].destination, flights[i].available_seats);
found = 1;
}
}
if (!found) {
printf("No flights found for the destination: %s\n", destination);
}
}
break;
case 4:
if (count == 0) {
printf("No flights available. Please add flights first.\n");
} else {
char flight_number[10];
int tickets;
printf("Enter flight number to cancel tickets: ");
scanf("%s", flight_number);
printf("Enter number of tickets to cancel: ");
scanf("%d", &tickets);
int found = 0;
```

```
for (int i = 0; i < count; i++) {
if (strcmp(flights[i].flight_number, flight_number) == 0) {
flights[i].available_seats += tickets;
printf("Tickets canceled successfully. Updated seats: %d\n", flights[i].available_seats);
found = 1;
break;
}
}
if (!foufound) {
printf("Flight not found.\n");
}
}
break;
case 5:
return 0;
default:
printf("Invalid choice! Please try again.\n");
break;
}
}
return 0;
}
```