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 <string.h>
#include <stdlib.h>
int main ()
{
   int n;
   printf("Enter the size of array: ");
   scanf("%d", &n);

   int *arr=(int *)malloc(n * sizeof(int));
   printf("Enter the array elements: ");
   for(int i=0; i<n; i++)
   {
      scanf("%d", arr+i);
   }

   char op = 'y';
   do
   {
      printf("\nResize array\ni -> increase the size of array\nd -> decrease the size of array\ne ->
exit\n");
      printf("Enter a option: ");
      scanf(" %c", &op);
```

```
switch(op)
{
   case 'i':
   {
      int s;
      printf("Enter the number of elements to increase: ");
      scanf("%d", &s);

      int new_size = n + s;
      arr = realloc(arr, new_size * sizeof(int));

      printf("Array elements after resizing are: ");
      for(int i=0; i<new_size; i++)
      {
         printf("%d ", arr[i]);
      }
   }
   break;
   case 'd':
   {
      int s;
      printf("Enter the number of elements to decrease: ");
      scanf("%d", &s);

      if (s >= n)
      {
         printf("Cannot decrease by %d elements. Size must remain positive.\n", s);
      }
      else
      {
```

```
                int new_size = n - s;
                arr = realloc(arr, new_size * sizeof(int));
                printf("Array elements after resizing are: ");
                for(int i=0; i<new_size; i++)
                {
                   printf("%d ", arr[i]);
                }
              }
            }
            break;
            case 'e':
            {
               printf("Exiting!!\n");
            }
            break;
            default: printf("Invalid option!!\n");
        }
    }while(op != 'e');

    free(arr);

    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>

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;
}

void my_strcat(char *str1, char *str2)
{

   int i=0;
   int j=strlen(str1);
   str1 = (char *)realloc(str1, 20 * sizeof(char));
   while(str2[i])
   {
      str1[j] = str2[i];
      i++;
      j++;
   }
}
```

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:
    - 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.

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

typedef struct node {
   int data;
   struct node *link;
} s_list;

int main() {
   int op = 0;
   s_list *head = NULL;

   do
   {
      printf("\nLinked list operations:\n");
      printf("1. Add a node to the list\n");
      printf("2. Delete a node from the list\n");
      printf("3. Display the list\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
```

```
scanf(" %d", &op);

switch (op)
{
   case 1:
   {
      int data;
      printf("Enter the data to be added: ");
      scanf("%d", &data);

      s_list *new_node = (s_list *)malloc(sizeof(s_list));
      if (!new_node)
      {
         printf("Memory allocation failed\n");
         break;
      }
      new_node->data = data;
      new_node->link = NULL;

      if (head == NULL)
      {
         head = new_node;
      }
      else
      {
         s_list *temp = head;
         while (temp->link != NULL)
         {
            temp = temp->link;
         }
```

```
        temp->link = new_node;
    }
    printf("New node is added successfully!\n");
}
break;

case 2:
{
    if (head == NULL)
    {
        printf("List is empty\n");
    }
    else if (head->link == NULL)
    { // Only one node in the list
        free(head);
        head = NULL;
    }
    else
    {
        s_list *temp = head;
        while (temp->link->link != NULL)
        {
            temp = temp->link;
        }
        free(temp->link);
        temp->link = NULL;
    }
    printf("Deleted Successfully!\n");
}
break;
```

```
        case 3:
        {
            if (head == NULL)
            {
                printf("List is empty\n");
            }
            else
            {
                s_list *temp = head;
                while (temp)
                {
                    printf("%d -> ", temp->data);
                    temp = temp->link;
                }
                printf("NULL\n");
            }
        }
        break;

        case 4:
            printf("Exiting!!\n");
            break;

        default:
            printf("Invalid option. Please try again.\n");
    }
} while (op != 4);

// Free all nodes before exiting
while (head)
{
```

```
    s_list *temp = head;
    head = head->link;
    free(temp);
  }

  return 0;
}on
```

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 6: 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:

o 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>

typedef struct Student
{
   char name[50];
   int roll_no;
   float marks;
} student;

void add_student(student s[], int n);
```

```
void display_student(student s[], int no);
void findStudent(student s[], int no);
void calculateAverage(student s[], int no);

int main()
{
   student s[100];
   int no = 0;
   int op = 0;

   printf("Student Record Management System\n");

   do
   {
      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("5. Exit\n");
      printf("Enter a choice: ");
      scanf(" %d", &op);

      switch (op)
      {
         case 1:
            add_student(s, no);
            no++;
            printf("Student data added successfully.\n");
            break;

         case 2:
```

```
            display_student(s, no);
            break;

        case 3:
            findStudent(s, no);
            break;

        case 4:
            calculateAverage(s, no);
            break;

        case 5:
            printf("Exiting!!\n");
            break;

        default:
            printf("Invalid option!!!\n");
      }
   } while (op != 5);

   return 0;
}

void add_student(student s[], int n)
{
   printf("Enter the name of the student: ");
   scanf(" %[^\n]", s[n].name);
   printf("Enter the roll no: ");
   scanf("%d", &s[n].roll_no);
   printf("Enter the marks: ");
   scanf("%f", &s[n].marks);
```

```
}

void display_student(student s[], int no)
{
   if (no == 0)
   {
      printf("No students available.\n");
      return;
   }

   printf("\nStudent Records:\n");
   for (int i = 0; i < no; i++)
   {
      printf("\nStudent %d:\n", i + 1);
      printf("Name: %s\nRoll No: %d\nMarks: %.2f\n", s[i].name, s[i].roll_no, s[i].marks);
   }
}

void findStudent(student s[], int no)
{
   if (no == 0)
   {
      printf("No students available.\n");
      return;
   }

   int roll_no;
   printf("Enter the roll number to search: ");
   scanf("%d", &roll_no);

   for (int i = 0; i < no; i++)
```

```
    {
        if (s[i].roll_no == roll_no)
        {
            printf("Student Found:\n");
            printf("Name: %s\nRoll No: %d\nMarks: %.2f\n", s[i].name, s[i].roll_no, s[i].marks);
            return;
        }
    }

    printf("Student with Roll No %d not found.\n", roll_no);
}

void calculateAverage(student s[], int no)
{
    if (no == 0)
    {
        printf("No students available.\n");
        return;
    }

    float total = 0.0;
    for (int i = 0; i < no; i++)
    {
        total += s[i].marks;
    }

    printf("Average Marks: %.2f\n", total / no);
}
```

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>

typedef struct Employee_details
{
   int emp_id;
   char name[50];
   float salary;
}Employees;

void add_employee(Employees [], int);
void update_salary(Employees[], int);
void display(Employees[], int);
void highest_salary(Employees[], int);

int main()
{
   Employees employee[100];
   int op = 0, count = 0;
   printf("Employee Management System\n");
```

```
do{
   printf("\n1.Add an employee.\n");
   printf("2.Update employee salary by ID.\n");
   printf("3.Display all employee details.\n");
   printf("4.Find and display details of the employee with the highest salary.\n");
   printf("5.Exit\n");
   printf("Enter your choice: ");
   scanf(" %d", &op);

   switch(op)
   {
      case 1:
      {
         add_employee(employee, count);
         count++;
         printf("Employee added successfully!\n");
      }
      break;
      case 2:
      {
         update_salary(employee, count);
      }
      break;
      case 3:
      {
         display(employee, count);
      }
      break;
      case 4:
      {
         highest_salary(employee, count);
```

```
        }
        break;
        case 5:
        {
          printf("Exiting from Employee Management System!!\n");
        }
        break;
        default:
        printf("Invalid option!! Please try again\n");
      }
   }while(op != 5);
}

void add_employee(Employees employee[], int i)
{
   printf("Enter the employee ID: ");
   scanf("%d", &employee[i].emp_id);
   printf("Enter the employee name: ");
   scanf(" %[^\n]", employee[i].name);
   printf("Enter the salary amount: ");
   scanf("%f", &employee[i].salary);
}

void update_salary(Employees employee[], int count)
{
   int id, found=0;
   printf("Enter the employee ID: ");
   scanf("%d", &id);
   for(int i=0; i<count; i++)
   {
      if(employee[i].emp_id == id)
```

```
        {
            printf("Enter the new salary amount of %s: ",employee[i].name);
            scanf("%f", &employee[i].salary);
            printf("Salary updated successfully\n");
            found=1;
            break;
        }

    }

    if(!found)
    {
        printf("Employee with ID %d not found.\n", id);
    }
}

void display(Employees employee[], int count)
{
    if (count == 0)
    {
        printf("No employees to display.\n");
        return;
    }

    printf("\nEmployee Details:\n");
    for (int i = 0; i < count; i++)
    {
        printf("ID: %d, Name: %s, Salary: %.2f\n", employee[i].emp_id, employee[i].name,
employee[i].salary);
    }
}
```

```
void highest_salary(Employees employee[], int count)
{
   if (count == 0)
   {
      printf("No employees to evaluate.\n");
      return;
   }

   int index = 0;
   for (int i = 1; i < count; i++)
   {
      if (employee[i].salary > employee[index].salary)
      {
         index = i;
      }
   }

   printf("\nEmployee with Highest Salary:\n");
   printf("ID: %d, Name: %s, Salary: %.2f\n",
         employee[index].emp_id,
         employee[index].name,
         employee[index].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>

typedef struct Book
{
   int book_id;
   char title[100];
   char author[50];
   int copies;
}book;



void addBook(book books[], int count);
void issueBook(book books[], int count);
void returnBook(book books[], int count);
void searchBook(book books[], int count);

int main()
```

```
{
    book books[100];
    int count = 0;
    int op = 0;

    do
    {
        printf("\nLibrary Management System\n");
        printf("1. Add Book\n");
        printf("2. Issue Book\n");
        printf("3. Return Book\n");
        printf("4. Search Book\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &op);

        switch (op)
        {
            case 1:
            {
                addBook(books, count);
                count++;
                printf("Book added successfully!\n");
            }
            break;
            case 2:
                issueBook(books, count);
                break;
            case 3:
                returnBook(books, count);
                break;
```

```
            case 4:
                searchBook(books, count);
                break;
            case 5:
                printf("Exiting program!!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (op != 5);

    return 0;
}



void addBook(book books[], int count)
{
    if (count >= 100)
    {
        printf("Library is full! Cannot add more books.\n");
        return;
    }

    printf("Enter Book ID: ");
    scanf("%d", &books[count].book_id);
    printf("Enter Book Title: ");
    scanf(" %[^\n]", books[count].title);
    printf("Enter Author Name: ");
    scanf(" %[^\n]", books[count].author);
    printf("Enter Number of Copies: ");
    scanf("%d", &books[count].copies);
```

```
}

void issueBook(book books[], int count)
{
   int id, found = 0;

   printf("Enter Book ID to issue: ");
   scanf("%d", &id);

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

   if (!found)
```

```
    {
        printf("Book with ID %d not found.\n", id);
    }
}


void returnBook(book books[], int count)
{
    int id, found = 0;

    printf("Enter Book ID to return: ");
    scanf("%d", &id);

    for (int i = 0; i < count; i++)
    {
        if (books[i].book_id == id)
        {
            found = 1;
            books[i].copies++;
            printf("Book \"%s\" returned successfully! Total copies: %d\n",
                  books[i].title, books[i].copies);
            break;
        }
    }

    if (!found) {
        printf("Book with ID %d not found.\n", id);
    }
}

void searchBook(book books[], int count)
```

```
{
    char query[100];
    int found = 0;

    printf("Enter Title or Author to search: ");
    scanf(" %[^\n]", query);

    printf("\nSearch Results:\n");
    for (int i = 0; i < count; i++)
    {
        if (strstr(books[i].title, query) || strstr(books[i].author, query))
        {
            printf("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 matching the query \"%s\".\n", query);
    }
}
```

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>

typedef struct Player
{
   char name[50];
   int matches;
   int runs;
   float average;
} Player;


void addPlayerDetails(Player players[], int n);
void calculateAverages(Player players[], int n);
void displayPlayerDetails(const Player players[], int n);
void findHighestAverage(const Player players[], int n);

int main()
{
```

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

    Player players[n];

    addPlayerDetails(players, n);
    calculateAverages(players, n);
    displayPlayerDetails(players, n);
    findHighestAverage(players, n);

    return 0;
}



void addPlayerDetails(Player players[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter 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);
        players[i].average = 0.0;
    }
}
```

```
void calculateAverages(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(const Player players[], int n)
{
   printf("\nPlayer Details:\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);
   }
}



void findHighestAverage(const Player players[], int n)
{
   if (n == 0)
   {
```

```
        printf("\nNo players to evaluate.\n");
        return;
    }

    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, Average: %.2f\n",players[highestIndex].name,
players[highestIndex].matches,players[highestIndex].runs, 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>

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



void inputStudentDetails(Student students[], int n);
void calculateGrades(Student students[], int n);
void displayStudentDetails(const Student students[], int n);
char assignGrade(float average);

int main()
{
   int n;
   printf("Enter the number of students: ");
```

```
    scanf("%d", &n);

    Student students[n];

    inputStudentDetails(students, n);
    calculateGrades(students, n);
    displayStudentDetails(students, n);

    return 0;
}


void inputStudentDetails(Student students[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter details for Student %d:\n", i + 1);
        printf("Roll Number: ");
        scanf("%d", &students[i].roll_no);
        printf("Name: ");
        scanf(" %[^\n]", students[i].name);
        printf("Enter marks for 5 subjects: ");
        for (int j = 0; j < 5; j++)
        {
            scanf("%f", &students[i].marks[j]);
        }
    }
}


void calculateGrades(Student students[], int n)
```

```
{
    for (int i = 0; i < n; i++)
    {
        float total = 0.0;
        for (int j = 0; j < 5; j++)
        {
            total += students[i].marks[j];
        }
        float average = total / 5;
        students[i].grade = assignGrade(average);
    }
}


char assignGrade(float average)
{
    if (average >= 90)
    {
        return 'A';
    }
    else if (average >= 80)
    {
        return 'B';
    }
    else if (average >= 70)
    {
        return 'C';
    }
    else if (average >= 60)
    {
        return 'D';
```

```
    }
    else
    {
        return 'F';
    }
}



void displayStudentDetails(const Student students[], int n)
{
    printf("\nStudent Details:\n");
    printf("Roll No\t\tName\t\tAverage\t\tGrade\n");
    for (int i = 0; i < n; i++)
    {
        float total = 0.0;
        for (int j = 0; j < 5; j++)
        {
            total += students[i].marks[j];
        }
        float average = total / 5;
        printf("%d\t\t%s\t\t%.2f\t\t%c\n", students[i].roll_no, students[i].name, average,
students[i].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>

typedef struct Flight
{
   char flight_number[10];
   char destination[50];
   int available_seats;
} Flight;



void addFlight(Flight flights[], int count);
void bookTicket(Flight flights[], int count);
void displayFlights(Flight flights[], int count);
void cancelTicket(Flight flights[], int count);

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

```
int count = 0;
int choice;

do {
    printf("\nFlight Reservation System\n");
    printf("1. Add Flight\n");
    printf("2. Book Ticket\n");
    printf("3. Display Flights by Destination\n");
    printf("4. Cancel Ticket\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
        case 1:
            addFlight(flights, count);
            count++;
            printf("Flight added successfully!\n");
            break;
        case 2:
            bookTicket(flights, count);
            break;
        case 3:
            displayFlights(flights, count);
            break;
        case 4:
            cancelTicket(flights, count);
            break;
        case 5:
            printf("Exiting system. Goodbye!\n");
```

```
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 5);

  return 0;
}



void addFlight(Flight flights[], int count)
{
  if (count >= 100)
  {
    printf("Cannot add more flights. System is full.\n");
    return;
  }

  printf("Enter Flight Number: ");
  scanf("%s", flights[count].flight_number);
  printf("Enter Destination: ");
  scanf(" %[^\n]", flights[count].destination);
  printf("Enter Number of Available Seats: ");
  scanf("%d", &flights[count].available_seats);
}

void bookTicket(Flight flights[], int count)
{
  char flight_number[10];
  int found = 0;
```

```
    printf("Enter Flight Number to book a ticket: ");
    scanf("%s", flight_number);

    for (int i = 0; i < count; 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 for Flight %s! Remaining seats: %d\n",
                    flights[i].flight_number, flights[i].available_seats);
            }
            else
            {
                printf("No seats available for Flight %s.\n", flights[i].flight_number);
            }
            break;
        }
    }

    if (!found)
    {
        printf("Flight with number %s not found.\n", flight_number);
    }
}

void displayFlights(Flight flights[], int count)
{
    char destination[50];
```

```
    int found = 0;

    printf("Enter Destination to search: ");
    scanf(" %[^\n]", destination);

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

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


void cancelTicket(Flight flights[], int count)
{
    char flight_number[10];
    int found = 0;

    printf("Enter Flight Number to cancel a ticket: ");
    scanf("%s", flight_number);
```

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

    if (!found)
    {
        printf("Flight with number %s not found.\n", flight_number);
    }
}
```