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>

void print_arr(int *arr,int size);

int main()

{

   int size;

   printf("Enter the initial size of array:");

   scanf("%d",&size);

   int *ptr = (int *)malloc(size * sizeof(int));

   printf("Enter the elements of array:");

   for(int i=0;i<size;i++)

   {

      scanf("%d",ptr+i);



   }

   char op;

   int resize;

   printf("Array elements before modification:");
```

```
print_arr(ptr,size);


do{


    printf("\nEnter the option:\ni->To increase size of array\nd->Decrease size of
array\ne->To exit\n");
    getchar();
    scanf("%c",&op);
     switch(op)
     {
        case 'i':
        printf("Enter the size to be increased:");
        scanf("%d",&resize);
        ptr = realloc(ptr,resize);
        printf("Array elements after incrementing size:");
        print_arr(ptr,resize);
        break;


        case 'd':
        printf("Enter the size to be increased:");
        scanf("%d",&resize);
        ptr = realloc(ptr,resize);
        printf("Array elements after decrementing size:");
        print_arr(ptr,resize);
        break;
```

```
            case 'e':
            printf("Exiting\n");
            break;

            default:
            printf("Invalid option.");
        }
    }while(op != 'e');
    free(ptr);
}

void print_arr(int *arr,int size)
{
    for(int i=0;i<size;i++)
    {
        printf("%d ",*(arr+i));

    }
}
```

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=NULL;
    char str2[10];
    printf("Enter the first string:");
    str1 = (char *)malloc(10 * sizeof(char));
    //scanf("%[^\n]",str1);
    fgets(str1, 10, stdin);
    getchar();
    printf("enter the second string:");
    scanf("%s",str2);
    printf("The string1 before concatenated is %s\n",str1);
    printf("The string2 before concatenated is %s\n",str2);
    str1 = realloc(str1,20);
    strcat(str1,str2);

    printf("The string1 after concatenated is %s",str1);
    free(str1);

}
```

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 **array;
   int rows, cols;

   printf("Enter the number of rows: ");
   scanf("%d", &rows);
   printf("Enter the number of columns: ");
   scanf("%d", &cols);


   array = (int **)malloc(rows * sizeof(int *));
   if (array == NULL)
   {
      printf("Memory allocation failed!\n");
      return 1;
   }


   for (int i = 0; i < rows; i++)
   {
      array[i] = (int *)malloc(cols * sizeof(int));
      if (array[i] == NULL)
      {
         printf("Memory allocation failed for row %d!\n", i);

         for (int j = 0; j < i; j++)
         {
            free(array[j]);
         }
         free(array);
```

```
            return 1;
        }
    }


    printf("Enter the elements of the 2D array:\n");
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            printf("Element [%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);

    return 0;
}
```

6.Problem statement :Students record

```
#include <stdio.h>


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

```
void addStd_details(struct student Student[], int *count);
void display_details(struct student Student[],int count);
void find_student(struct student Student[],int count);
void calculate_avg_Marks(struct student Student[],int count);

int main()
{
   struct student Student[100];
   int count=0;
   int op;
   do{
      printf("Choose appropriate option:\n1.Add Student\n2.Display all student
details\n3.Find student by roll number\n4.Calculate average marks\n5.Exit\n");

      scanf("%d",&op);
      switch(op)
      {
         case 1:
         addStd_details(Student,&count);
         break;
         case 2:
         display_details(Student,count);
         break;

         case 3:
         find_student(Student,count);
         break;

         case 4:
         calculate_avg_Marks(Student, count);
         break;

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

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

return 0;
}



void addStd_details(struct student Student[], int *count)
```

```
{
   printf("Enter student name: ");
   scanf(" %[^\n]", Student[*count].name);
   printf("Enter roll number: ");
   scanf("%d", &Student[*count].roll_no);
   printf("Enter marks: ");
   scanf("%f", &Student[*count].marks);
   (*count)++;
   printf("Student added successfully!\n");
}


void display_details(struct student Student[],int count)
{
   if (count == 0) {
      printf("No students available.\n");
      return;
   }
   printf("\nStudent Records:\n");
   for (int i = 0; i < count; i++)
   {
      printf("Name: %s, Roll No: %d, Marks: %.2f\n", Student[i].name,
Student[i].roll_no, Student[i].marks);
   }

}

void find_student(struct student Student[],int count)
{
    if (count == 0)
    {
      printf("No students available.\n");
      return;
   }
   int roll_no;
   printf("Enter roll number to search: ");
   scanf("%d", &roll_no);
   for (int i = 0; i < count; i++)
   {
      if (Student[i].roll_no == roll_no)
      {
         printf("Name: %s, Roll No: %d, Marks: %.2f\n", Student[i].name,
Student[i].roll_no, Student[i].marks);
         return;
      }
   }
   printf("\n%d roll no. is not available\n",roll_no);
```

```
}

void calculate_avg_Marks(struct student Student[],int count)
{

   if (count == 0)
   {
      printf("No students available.\n");
      return;
   }
   float total = 0;
   for (int i = 0; i < count; i++)
   {
      total += Student[i].marks;
   }
   printf("Average Marks: %.2f\n", total / count);

}
```

Problem 7: Employee Management System

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

Description:

1. Define a structure Employee with fields:
    1. int emp_id: Employee ID
    2. char name[50]: Employee name
    3. float salary: Employee salary
2. Write a menu-driven program to:

    1. Add an employee.
    2. Update employee salary by ID.
    3. Display all employee details.
    4. Find and display details of the employee with the highest salary.

```
#include <stdio.h>


struct Employee{

   int emp_id;
```

```
    char name[50];
    float salary;
};
void add_emp(struct Employee employee[],int *count);
void update_emp_sal(struct Employee employee[],int count);
void disp_details(struct Employee employee[],int count);


int main()
{
    struct Employee employee[100];
    int count = 0;
    int op;
    do{
        printf("Choose the appropriate option:\n1.Add an employee.\n2.Update
employee salary by ID.\n3.Display all employee details.\n4.Find and display details of
the employee with the highest salary.\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
            add_emp(employee,&count);
            break;


            case 2:
            update_emp_sal(employee,count);
            break;
```

```
case 3:
disp_details(employee,count);
break;



case 4:
if (count == 0)
{
    printf("No employee details available\n");
}
else
{
    int maxIndex = 0;
    for (int i = 1; i < count; i++)
    {
        if (employee[i].salary > employee[maxIndex].salary)
        {
            maxIndex = i;
        }
    }
    printf("\nEmployee with the highest salary:\n");
    printf("Employee ID: %d\n", employee[maxIndex].emp_id);
    printf("Employee Name: %s\n", employee[maxIndex].name);
    printf("Employee Salary: %.2f\n", employee[maxIndex].salary);
```

```
            }
            break;

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

        }
    }while(op != 5);
}

void add_emp(struct Employee employee[],int *count)
{
    printf("Enter employee ID:");
    scanf("%d",&employee[*count].emp_id);
    printf("\nEnter the name of employee:");
    getchar();
    scanf("%[^\n]",employee[*count].name);
    printf("\nEnter the salary:");
    scanf("%f",&employee[*count].salary);
    (*count)++;
    printf("Entered the details of %d employee successfully!\n",*count);

}
```

```
void update_emp_sal(struct Employee employee[],int count)
{
   if(count == 0)
   {
      printf("No employee details available\n");
   }
   else
   {
     int id;
     printf("\nEnter the employee Id to be updated:");
     scanf("%d",&id);
     float upd_salary;
     printf("\nEnter the salary to be updated:");
     scanf("%f",&upd_salary);
     int found = 0;
     for(int i=0;i<count;i++)
     {
        if(id == employee[i].emp_id)
        {
           employee[i].salary = upd_salary;
           printf("\nUpdated salary successfully!\n");
           found = 1;
           break;
        }
```

```
        }
        if (!found)
        {
            printf("\nNo such employee ID available!\n");
        }

    }

}

void disp_details(struct Employee employee[],int count)
{
     if(count == 0)
    {
        printf("No employee details available\n");
    }
    else
    {
        for(int i=0;i<count;i++)
        {
            printf("\nEmployee Id :%d\tEmployee name : %s\tEmployee
salary : %.2f\t\n",employee[i].emp_id,employee[i].name,employee[i].salary);

        }
    }
}
```

Problem 8: Library Management System

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

Description:

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

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

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

void add_book(struct Book library[], int *count);
void issue_book(struct Book library[], int count);
void return_book(struct Book library[], int count);
void search_book(struct Book library[], int count);
```

```
int main() {
    struct Book library[100];
    int count = 0;
    int op;

    do {
        printf("\nLibrary Management System\n1. Add Book\n2. Issue Book\n3. Return Book\n4. Search Book\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &op);

        switch (op) {
            case 1:
                add_book(library, &count);
                break;
            case 2:
                issue_book(library, count);
                break;
            case 3:
                return_book(library, count);
                break;
            case 4:
                search_book(library, count);
                break;
            case 5:
```

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

return 0;
}



void add_book(struct Book library[], int *count)
{
    printf("Enter Book ID: ");
    scanf("%d", &library[*count].book_id);

    printf("Enter Book Title: ");
    getchar(); // Consume the newline character left by scanf
    scanf("%[^\n]", library[*count].title);

    printf("Enter Author Name: ");
    getchar();
    scanf("%[^\n]", library[*count].author);

    printf("Enter Number of Copies: ");
```

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

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


void issue_book(struct Book library[], int count)
{
    if (count == 0)
    {
        printf("No books available in the library.\n");
        return;
    }

    int book_id, found = 0;
    printf("Enter Book ID to issue: ");
    scanf("%d", &book_id);

    for (int i = 0; i < count; i++)
    {
        if (library[i].book_id == book_id)
        {
            found = 1;
            if (library[i].copies > 0)
```

```
            {
                library[i].copies--;
                printf("Book issued successfully!\n");
            }
            else
            {
                printf("No copies available for this book.\n");
            }
            break;
        }
    }

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



void return_book(struct Book library[], int count)
{
    if (count == 0)
    {
        printf("No books available in the library.\n");
        return;
    }
```

```
    int book_id, found = 0;
    printf("Enter Book ID to return: ");
    scanf("%d", &book_id);

    for (int i = 0; i < count; i++)
    {
        if (library[i].book_id == book_id)
        {
            found = 1;
            library[i].copies++;
            printf("Book returned successfully!\n");
            break;
        }
    }

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


void search_book(struct Book library[], int count) {
    if (count == 0)
```

```
{
    printf("No books available in the library.\n");
    return;
}

char keyword[100];
int found = 0;
printf("Enter title or author to search: ");
getchar();
scanf("%[^\n]", keyword);

for (int i = 0; i < count; i++)
{
    if (strcmp(library[i].title, keyword) || strcmp(library[i].author, keyword))
    {
        found = 1;
        printf("\nBook ID: %d\nTitle: %s\nAuthor: %s\nCopies Available: %d\n",
               library[i].book_id, library[i].title, library[i].author, library[i].copies);
    }
}

if (!found)
{
    printf("No books found matching the keyword '%s'.\n", keyword);
}
```

}

Problem 9: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. Define a structure Player with fields:
    1. char name[50]: Player name
    2. int matches: Number of matches played
    3. int runs: Total runs scored
    4. float average: Batting average
2. Write a program to:
    1. Input details for n players.
    2. Calculate and display the batting average for each player.
    3. Find and display the player with the highest batting average.

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

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

void input_details(struct Player players[], int n);
void calculate_average(struct Player players[], int n);
void display_highest_average(struct Player players[], int n);
```

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

    struct Player players[n];

    input_details(players, n);
    calculate_average(players, n);
    display_highest_average(players, n);

    return 0;
}



void input_details(struct Player players[], int n)
{
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for player %d:\n", i + 1);
        printf("Name of player: ");
        getchar();
        scanf("%[^\n]", players[i].name);
        printf("Number of matches played: ");
        scanf("%d", &players[i].matches);
        printf("Total runs scored: ");
```

```
        scanf("%d", &players[i].runs);
        players[i].average = 0.0;
    }
}

void calculate_average(struct Player players[], int n)
{
    printf("\nPlayer Details with Batting Average:\n");
    for (int i = 0; i < n; i++)
    {

        if (players[i].matches > 0) {
            players[i].average = (float)players[i].runs / players[i].matches;
        } else
        {
            players[i].average = 0.0;
        }
        printf("Player: %s | Matches: %d | Runs: %d | Average: %.2f\n",
               players[i].name, players[i].matches, players[i].runs, players[i].average);
    }
}



void display_highest_average(struct Player players[], int n) {
    if (n == 0)
```

```
    {
        printf("\nNo players available.\n");
        return;
    }

    int maxIndex = 0;
    for (int i = 1; i < n; i++)
    {
        if (players[i].average > players[maxIndex].average) {
            maxIndex = i;
        }
    }

    printf("\nPlayer with the highest batting average:\n");
    printf("Name: %s\n", players[maxIndex].name);
    printf("Matches: %d\n", players[maxIndex].matches);
    printf("Runs: %d\n", players[maxIndex].runs);
    printf("Average: %.2f\n", players[maxIndex].average);
}
```

Problem 10: Student Grading System

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

Description:

1. Define a structure Student with fields:
    1. int roll_no: Roll number
    2. char name[50]: Student name

3. float marks[5]: Marks in 5 subjects
4. char grade: Grade based on the average marks

2. Write a program to:

    1. Input details of n students.
    2. Calculate the average marks and assign grades (A, B, C, etc.).
    3. Display details of students along with their grades.

```
#include <stdio.h>

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

void input_details(struct Student students[], int n);
void calculate_grades(struct Student students[], int n);
void display_details(struct Student students[], int n);

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

```
    struct Student students[n];

    input_details(students, n);
    calculate_grades(students, n);
    display_details(students, n);

    return 0;
}


void input_details(struct 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: ");
        getchar();
        scanf("%[^\n]", students[i].name);

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

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



void calculate_grades(struct 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 / 5.0;
        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 display_details(struct Student students[], int n) {
    printf("\nStudent Details:\n");
    printf("Roll No\tName\t\t\tMarks (5 subjects)\tGrade\n");
    printf("---------------------------------------------------------------\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%-20s\t", students[i].roll_no, students[i].name);
        for (int j = 0; j < 5; j++)
        {
```

```
            printf("%.2f ", students[i].marks[j]);
        }
        printf("\t%c\n", students[i].grade);
    }
}
```

Problem 11: Flight Reservation System

Objective: Simulate a simple flight reservation system using structures.

Description:

1. Define a structure Flight with fields:
    1. char flight_number[10]: Flight number
    2. char destination[50]: Destination city
    3. int available_seats: Number of available seats
2. Write a program to:
    1. Add flights to the system.
    2. Book tickets for a flight, reducing available seats accordingly.
    3. Display the flight details based on destination.
    4. Cancel tickets, increasing the number of available seats.

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

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


void add_flights(struct Flight flights[], int *count);
void book_tickets(struct Flight flights[], int count);
void display_flights_by_destination(struct Flight flights[], int count);
void cancel_tickets(struct Flight flights[], int count);

int main()
{
    struct Flight flights[100];
    int count = 0;
```

```
    int choice;

    do {
        printf("\nFlight Reservation System\n1. Add Flight\n2. Book Tickets\n3. Display
Flights by Destination\n4. Cancel Tickets\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                add_flights(flights, &count);
                break;

            case 2:
                book_tickets(flights, count);
                break;

            case 3:
                display_flights_by_destination(flights, count);
                break;

            case 4:
                cancel_tickets(flights, count);
                break;

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

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

    return 0;
}


void add_flights(struct Flight flights[], int *count)
{
    printf("\nEnter details for flight %d:\n", *count + 1);
    printf("Flight number: ");
    scanf("%s", flights[*count].flight_number);
    printf("Destination: ");
    getchar();
    scanf("%[^\n]", flights[*count].destination);
```

```
    printf("Available seats: ");
    scanf("%d", &flights[*count].available_seats);

    (*count)++;
    printf("Flight added successfully!\n");
}


void book_tickets(struct Flight flights[], int count)
{
    if (count == 0)
    {
        printf("No flights available!\n");
        return;
    }

    char flight_number[10];
    int tickets;
    printf("\nEnter flight number: ");
    scanf("%s", flight_number);
    printf("Enter number of tickets to book: ");
    scanf("%d", &tickets);

    for (int i = 0; i < count; i++)
    {
        if (strcmp(flights[i].flight_number, flight_number) == 0)
        {
            if (tickets <= flights[i].available_seats)
            {
                flights[i].available_seats -= tickets;
                printf("Tickets booked successfully! Remaining seats: %d\n",
flights[i].available_seats);
            } else
            {
                printf("Not enough seats available! Only %d seats left.\n",
flights[i].available_seats);
            }
            return;
        }
    }
    printf("Flight not found!\n");
}


void display_flights_by_destination(struct Flight flights[], int count)
{
    if (count == 0)
```

```
    {
        printf("No flights available!\n");
        return;
    }

    char destination[50];
    printf("\nEnter destination: ");
    getchar();
    scanf("%[^\n]", destination);

    printf("\nFlights to %s:\n", destination);
    printf("Flight Number\tAvailable Seats\n");
    printf("---------------------------------\n");
    int found = 0;
    for (int i = 0; i < count; i++)
    {
        if (strcmp(flights[i].destination, destination) == 0)
        {
            printf("%s\t\t%d\n", flights[i].flight_number, flights[i].available_seats);
            found = 1;
        }
    }
    if (!found)
    {
        printf("No flights to %s found.\n", destination);
    }
}


void cancel_tickets(struct Flight flights[], int count)
{
    if (count == 0) {
        printf("No flights available!\n");
        return;
    }

    char flight_number[10];
    int tickets;
    printf("\nEnter flight number: ");
    scanf("%s", flight_number);
    printf("Enter number of tickets to cancel: ");
    scanf("%d", &tickets);

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

```
            flights[i].available_seats += tickets;
            printf("Tickets cancelled successfully! Available seats: %d\n",
flights[i].available_seats);
            return;
        }
    }
    printf("Flight not found!\n");
}
```