PHASE 1 DAY 12

CALLOC

```
#include <stdio.h>
#include<stdlib.h>
int main()
{

    int *ptr=NULL;
    int n;
    printf("enter the no of integers that will be stored:");
    scanf("%d",&n);
    ptr=(int*)calloc(n,sizeof(int));

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

REALLOC

```
#include <stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    char *str=NULL;
    str=(char*)malloc(15);
    strcpy(str,"jason");
    printf("string=%s ,address=%u\n",str,str);
   //reallocating memory
   str=(char*)realloc(str,25);
   strcat(str,".com");
    printf("string=%s ,address=%u\n",str,str);

free(str);
  return 0;
}
```

<u>output</u>

```
string=jason ,address=2184618656
string=jason.com ,address=2184619728
```

<u>DOUBLE POINTER</u>

```
#include <stdio.h>
//#include<stdlib.h>
int main()
{

   int **ipp;
   int i=4,j=5,k=6;
   int *ip1,*ip2;

   ip1=&i;
   ip2=&j;

   ipp=&ip1;

   printf("001 i=%d\n",*ip1);
   printf("002 i=%d\n",**ipp);

   **ipp=10;
   printf("003  modified i=%d",**ipp);

    return 0;
}
```

Problem 1: Dynamic Array Resizing

Objective: Write a program to dynamically allocate an integer array and allow the user to resize it.

Description:

1. **The program should ask the user to enter the initial size of the array.**
2. **Allocate memory using malloc.**

3. **Allow the user to enter elements into the array.**
4. **Provide an option to increase or decrease the size of the array. Use realloc to adjust the size.**
5. **Print the elements of the array after each resizing operation.**

Answer:

```
#include <stdio.h>

#include<stdlib.h>

int main()


{

    int size;

    int *ptr=NULL;

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

    scanf("%d",&size);

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

          if(ptr==NULL)

     {

     printf("Memory not allocated properly\n");

     }

     else

     {

         printf("enter elements to the array:");

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

         {
            scanf("%d",&ptr[i]);
         }
        printf("Array elements:");

          for(int i=0;i<size;i++)
         {
            printf("%d ",ptr[i]);
         }

        printf("\nDo you want to increase or decrease size of array?\nEnter
1->Increase\n2->Decrease\n3.Exit");
        int opt;
        scanf("%d",&opt);

        switch(opt)
        {
           case 1:
           {
               int newsize;
               printf("Enter the new size of array:");
               scanf("%d",&newsize);

               ptr= (int*)realloc(ptr,newsize*sizeof(int));
```

```
 if(ptr==NULL)
 {
  printf("Memory not allocated properly\n");


 }
else
{
 if(newsize>size)
 {
     printf("Enter %d more elements:",newsize-size);
     for(int i=size;i<newsize;i++)
     {
         scanf("%d",&ptr[i]);
     }


     printf("Array elements after resizing:");


     for(int i=0;i<newsize;i++)
     {
        printf("%d ",ptr[i]);
     }
 }
    free(ptr);
```

```
        ptr=NULL;
      break;


   }
}
case 2:
{
     int newsize;
    printf("Enter the new size of array:");
    scanf("%d",&newsize);


    ptr= (int*)realloc(ptr,newsize*sizeof(int));


    if(ptr==NULL)
    {
     printf("Memory not allocated properly\n");


    }
   else
   {
        printf("Array elements after reducing:");


        for(int i=0;i<newsize;i++)
```

```
                    {
                       printf("%d ",ptr[i]);
                    }
                }
                   free(ptr);
                   ptr=NULL;
                  break;
               }

            case 3:
              {
              free(ptr);
              ptr=NULL;
              exit(0);
              break;
              }
            default:
            printf("Invalid");
    }

}
    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.**

Answer:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



int main() {

    char *str1, *str2, *result;

    int len1, len2;

    printf("Enter the first string: ");



    str1 = (char *)malloc(100 * sizeof(char));



    if (str1 == NULL)

    {

        printf("Memory allocation failed\n");
    return 1;
}

scanf("%s", str1);


printf("Enter the second string: ");
str2 = (char *)malloc(100 * sizeof(char));

if (str2 == NULL)
{
    printf("Memory allocation failed\n");
    free(str1);
    return 1;
}
scanf("%s", str2);

len1 = strlen(str1);
len2 = strlen(str2);

result = (char *)realloc(str1, (len1 + len2 + 1) * sizeof(char));

if (result == NULL)
```

```
    {
        printf("Memory reallocation failed\n");
        free(str1);
        free(str2);
        return 1;
    }

    // Concatenate strings
    strcat(result, str2);
    printf("Concatenated String: %s\n", result);

    free(result);
    free(str2);
    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.**

Answer:

```
#include <stdio.h>

#include <stdlib.h>


 struct

 {

    int row;

    int col;

    int value;

} Element;


int main()

{

    int m, n, i, j, count = 0;

    int **matrix;

    Element *sparse;


    printf("Enter the number of rows and cols: ");

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


    matrix = (int **)malloc(m * sizeof(int *));

    for (i = 0; i < m; i++)

        matrix[i] = (int *)malloc(n * sizeof(int));
    printf("Enter the matrix elements:\n");
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
            if (matrix[i][j] != 0)
                count++;
        }

    sparse = (Element *)malloc(count * sizeof(Element));

    count = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (matrix[i][j] != 0) {
                sparse[count].row = i;
                sparse[count].col = j;
                sparse[count].value = matrix[i][j];
                count++;
            }

    printf("Sparse Matrix:\n");
    for (i = 0; i < count; i++)
        printf("Row: %d, Column: %d, Value: %d\n", sparse[i].row, sparse[i].col,
sparse[i].value);
```

```
    for (i = 0; i < m; i++)

        free(matrix[i]);


    free(matrix);

    free(sparse);


    return 0;

}
```

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.**

   Answer:

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

    int **array;

    int rows, cols, i, j;
    printf("Enter the number of rows and columns: ");

    scanf("%d %d", &rows, &cols);

    array = (int **)malloc(rows * sizeof(int *));

    for (i = 0; i < rows; i++)

        array[i] = (int *)malloc(cols * sizeof(int));

    printf("Enter the values of the array:\n");

    for (i = 0; i < rows; i++)

        for (j = 0; j < cols; j++)

            scanf("%d", &array[i][j]);

    printf("Matrix:\n");

    for (i = 0; i < rows; i++) {

        for (j = 0; j < cols; j++)

            printf("%d ", array[i][j]);

        printf("\n");

    }

    for (i = 0; i < rows; i++)

        free(array[i]);

    free(array);

    return 0;

}
```

STUDENT RECORD(STRUCTURES)

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


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

struct student students[5];
int stud_count = 0;

void Add_student();
void display();
void find_student();
float calculate_average();

int main()
{
    int choice;

    while (1)
    {
        printf("\nMenu:\n1. Add student\n2. Display all students\n3. Find student by roll
number\n4. Calculate average marks\n5. Exit\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                Add_student();
                break;
            case 2:
                display();
                break;
            case 3:
                find_student();
                break;
            case 4:
              float avg= calculate_average();
              printf("Average marks of all students: %.2f\n", avg);
                break;
            case 5:
                exit(0);
                break;
            default:
                printf("Invalid\n");
        }
    }
    return 0;
}

void Add_student()
{

    struct student new_student;
    printf("Enter name: ");
    scanf(" %[^\n]", new_student.name);
    printf("Enter roll number: ");
    scanf("%d", &new_student.roll_no);
    printf("Enter marks: ");
    scanf("%f", &new_student.marks);

    students[stud_count] = new_student;
    stud_count++;
    printf("Student added successfully!\n");
}

void display()
{
    if (stud_count == 0)
    {
        printf("No students to display.\n");
        return;
    }

    printf("\nStudent Records:\n");

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

void find_student()
{
    if (stud_count == 0)
{
        printf("No students to search.\n");
        return;
    }

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

    for (int i = 0; i < stud_count; i++)
    {
        if (students[i].roll_no == roll_no)
        {
            printf("Student Found:\n");
            printf("Name: %s\n", students[i].name);
            printf("Roll Number: %d\n", students[i].roll_no);
            printf("Marks: %.2f\n", students[i].marks);
            return;
        }
    }
    printf("No student found with roll number %d.\n", roll_no);
}

float calculate_average()
```

```
{
   if (stud_count == 0)

   {
      printf("No students to calculate average marks.\n");
      return 0;
   }

   float total_marks = 0.0;
   for (int i = 0; i < stud_count; i++)
   {
      total_marks += students[i].marks;
   }

   float average = total_marks / stud_count;
   return average;
}
```

2.

```
#include <stdio.h>
 struct student

{
        char name[50];
        int rollNumber;
        float marks;
 };

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;
}
```

3. Compounded arrays

```
//compounded literals
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Coordinate
{
 int x;
  int y;
};

void printCoordinate(struct Coordinate);

int main()
{

    printCoordinate((struct Coordinate){5, 6});

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

    printCoordinate(pointA);*/

    return 0;

}

void printCoordinate(struct Coordinate temp){

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

}
```

4.Coordinates

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

```
 struct Coordinate

{
          int x;
          int y;
 };

int main(){

          struct Coordinate Pnt[5];

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

          for(int i = 0; i < 5; i++){
          printf("Diaply the Coordinates at index %d is (%d,%d) \n",i,Pnt[i].x,Pnt[i].y);
          printf("\n");
          }

          return 0;
 }
```

5.#include <stdio.h>

```
struct Month{
          int noOfDays;
          char name[3];
 };

 int main(){
          struct Month allMonths[12];

          for(int i = 0; i < 12; i++){
          printf("Enter The Month Name and the no. of days associated with that month");
          scanf("%s  %d", allMonths[i].name, &allMonths[i].noOfDays);
          printf("\n");
          }
          for(int j = 0; j < 12; j++){
          printf("Name of the Month = %s having %d
\n",allMonths[j].name,allMonths[j].noOfDays);
          }
```

6.

```
#include <stdio.h>

 struct currentDate{
          int day;
          int month;
          int year;
 };
 struct currentTime{
          int sec;
          int min;
          int hours;
 };



struct CDateTime{
          struct currentDate d1;
          struct currentTime t1;
 };



int main(){

          struct CDateTime dt = {{21, 11, 2024}, {51, 01, 17}};



          printf("Current Date = %d-%d-%d \n",dt.d1.day,dt.d1.month,dt.d1.year);
          printf("Current Time = %d-%d-%d \n",dt.t1.sec,dt.t1.min,dt.t1.hours);

          return 0;
 }
```

Problem 1: Employee Management System

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

Description:

1. **Define a structure Employee with fields:**
    - **int emp_id: Employee ID**
    - **char name[50]: Employee name**
    - **float salary: Employee salary**
2. **Write a menu-driven program to:**
    - **Add an employee.**
    - **Update employee salary by ID.**
    - **Display all employee details.**
    - **Find and display details of the employee with the highest salary.**

Answer

```
#include <stdio.h>

#include <string.h>



struct Employee

{

    int emp_id;

    char name[50];

    float salary;

};



void addEmployee(struct Employee employees[], int *count) {

    printf("Enter Employee ID: ");

    scanf("%d", &employees[*count].emp_id);

    printf("Enter Employee Name: ");
    scanf(" %[^\n]", employees[*count].name);
    printf("Enter Employee Salary: ");
    scanf("%f", &employees[*count].salary);
    (*count)++;
}

void updateSalary(struct Employee employees[], int count)
{
    int id, found = 0;
    float newSalary;
    printf("Enter Employee ID to update salary: ");
    scanf("%d", &id);
    for (int i = 0; i < count; i++) {
        if (employees[i].emp_id == id) {
            printf("Enter New Salary: ");
            scanf("%f", &newSalary);
            employees[i].salary = newSalary;
            found = 1;
            break;
        }
    }
    if (!found)
    printf("Employee not found!\n");
```

```
}

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

void highestSalary(struct Employee employees[], int count)
{

    int max = 0;
    for (int i = 1; i < count; i++)
    {
        if (employees[i].salary > employees[max].salary)
        {
            max= i;
        }
    }
    printf("Employee with Highest Salary:\n");
    printf("ID: %d, Name: %s, Salary: %.2f\n",employees[max].emp_id,
employees[max].name, employees[max].salary);

}



int main()

{

    struct Employee employees[100];

    int count = 0, choice;

    while (1)

    {

        printf("\n1. Add Employee\n2. Update Salary\n3. Display All\n4. Highest Salary\n5.
Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                addEmployee(employees, &count);

                break;

            case 2:

                updateSalary(employees, count);

                break;

            case 3:

                displayAll(employees, count);
                break;

            case 4:

                highestSalary(employees, count);

                break;

            case 5:

                exit(0);

            default:

                printf("Invalid\n");

        }

    }

}
```

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>

void searchBook(struct Book library[], int count);

void addBook(struct Book library[], int *count) ;

void issueBook(struct Book library[], int count) ;

void returnBook(struct Book library[], int count);

struct Book {

    int book_id;

    char title[100];

    char author[50];

    int copies;

};

int main()

 {

    struct Book library[100];

    int count = 0, choice;

    while (1)

 {

        printf("\n1. Add Book\n2. Issue Book\n3. Return Book\n4. Search Book\n5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

 {
```

```
            case 1:
                addBook(library, &count);
                break;
            case 2:
                issueBook(library, count);
                break;
            case 3:
                returnBook(library, count);
                break;
            case 4:
                searchBook(library, count);
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid\n");
        }
    }
}
void addBook(struct Book library[], int *count)
{
    printf("Enter Book ID: ");
    scanf("%d", &library[*count].book_id);
    printf("Enter Book Title: ");

    scanf(" %[^\n]", library[*count].title);

    printf("Enter Author Name: ");

    scanf(" %[^\n]", library[*count].author);

    printf("Enter Number of Copies: ");

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

    (*count)++;

}


void issueBook(struct Book library[], int count)

{

    int id, found = 0;

    printf("Enter Book ID to issue: ");

    scanf("%d", &id);

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

        if (library[i].book_id == id && library[i].copies > 0) {

            library[i].copies--;

            printf("Book issued successfully.\n");

            found = 1;

            break;

        }

    }

    if (!found) printf("Book not found or out of stock.\n");
```

```
}


void returnBook(struct Book library[], int count)
 {
    int id, found = 0;
    printf("Enter Book ID to return: ");
    scanf("%d", &id);
    for (int i = 0; i < count; i++) {
        if (library[i].book_id == id) {
            library[i].copies++;
            printf("Book returned successfully.\n");
            found = 1;
            break;
        }
    }
    if (!found) printf("Book not found.\n");
}


void searchBook(struct Book library[], int count)
 {
    char query[100];
    printf("Enter Book Title or Author to search: ");
    scanf(" %[^\n]", query);
    for (int i = 0; i < count; i++) {
        if (strstr(library[i].title, query) || strstr(library[i].author, query)) {
            printf("ID: %d, Title: %s, Author: %s, Copies: %d\n",
                   library[i].book_id, library[i].title, library[i].author, library[i].copies);
        }
    }
}
```

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.**

answer:

```
#include <stdio.h>
#include <string.h>
void highestAverage(struct Player players[], int n);
void displayPlayers(struct Player players[], int n) ;
void inputPlayers(struct Player players[], int n);
struct Player {
    char name[50];

    int matches;

    int runs;

    float average;

};

int main()

 {

    struct Player players[100];

    int n;

    printf("Enter number of players: ");

    scanf("%d", &n);


    inputPlayers(players, n);

    displayPlayers(players, n);

    highestAverage(players, n);


    return 0;

}


void inputPlayers(struct Player players[], int n)

 {

    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 Scored: ");

        scanf("%d", &players[i].runs);

        players[i].average = (players[i].matches > 0) ? (float)players[i].runs /
players[i].matches : 0.0;

    }

}


void displayPlayers(struct Player players[], int n)

{

    printf("\nPlayer Details:\n");

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

        printf("Name: %s, Matches: %d, Runs: %d, Batting Average: %.2f\n",

               players[i].name, players[i].matches, players[i].runs, players[i].average);

    }

}


void highestAverage(struct Player players[], int n)

 {

    if (n == 0) {

        printf("No 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, Matches: %d, Runs: %d, Batting Average: %.2f\n",

           players[maxIndex].name, players[maxIndex].matches, players[maxIndex].runs,
players[maxIndex].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>
```

```
struct Student {
    int roll_no;
    char name[50];
    float marks[5];
    char grade;
};
void inputStudents(struct Student students[], int n);
void displayStudents(struct Student students[], int n);
int main()
{
    struct Student students[100];
    int n;
    printf("Enter number of students: ");
    scanf("%d", &n);

    inputStudents(students, n);
    displayStudents(students, n);

    return 0;
}

void inputStudents(struct Student students[], int n)
```

```
{
    for (int i = 0; i < n; i++) {
        printf("Enter 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 in 5 subjects:\n");
        for (int j = 0; j < 5; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%f", &students[i].marks[j]);
        }
    }
}

float calculateAverage(float marks[]) {
    float sum = 0;
    for (int i = 0; i < 5; i++) {
        sum += marks[i];
    }
    return sum / 5.0;
}
```

```
char calculateGrade(float average) {
    if (average >= 90) return 'A';
    if (average >= 75) return 'B';
    if (average >= 60) return 'C';
    if (average >= 50) return 'D';
    return 'F';
}

void displayStudents(struct Student students[], int n)
 {
    printf("\nStudent Details:\n");
    for (int i = 0; i < n; i++) {
        float average = calculateAverage(students[i].marks);
        students[i].grade = calculateGrade(average);
        printf("Roll Number: %d, Name: %s, Average: %.2f, Grade: %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.**

Answer:

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

struct Flight {
   char flight_number[10];
   char destination[50];
   int available_seats;
};
void cancelTicket(struct Flight flights[], int count);
void displayByDestination(struct Flight flights[], int count);
void bookTicket(struct Flight flights[], int count);
void addFlight(struct Flight flights[], int *count);

int main()
{
   struct Flight flights[100];
   int count = 0, choice;
   while (1) {
      printf("\n1. Add Flight\n2. Book Ticket\n3. Cancel Ticket\n4. Display by
Destination\n5. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            addFlight(flights, &count);
            break;
         case 2:
                bookTicket(flights, count);
                break;
            case 3:
                cancelTicket(flights, count);
                break;
            case 4:
                displayByDestination(flights, count);
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
}

void addFlight(struct Flight flights[], int *count)
 {
    printf("Enter Flight Number: ");
    scanf(" %[^\n]", flights[*count].flight_number);
    printf("Enter Destination: ");
    scanf(" %[^\n]", flights[*count].destination);
    printf("Enter Available Seats: ");
    scanf("%d", &flights[*count].available_seats);
    (*count)++;
}

void bookTicket(struct Flight flights[], int count)
 {
    char flightNumber[10];
    int found = 0;
    printf("Enter Flight Number to book: ");
    scanf(" %[^\n]", flightNumber);
    for (int i = 0; i < count; i++) {
        if (strcmp(flights[i].flight_number, flightNumber) == 0) {
            if (flights[i].available_seats > 0) {
                flights[i].available_seats--;
                printf("Ticket booked successfully.\n");
            } else {
                printf("No seats available.\n");
            }
            found = 1;
            break;
        }
    }
    if (!found) printf("Flight not found.\n");
}

void cancelTicket(struct Flight flights[], int count)
 {
    char flightNumber[10];
    int found = 0;
    printf("Enter Flight Number to cancel ticket: ");
    scanf(" %[^\n]", flightNumber);
    for (int i = 0; i < count; i++) {
        if (strcmp(flights[i].flight_number, flightNumber) == 0) {
            flights[i].available_seats++;
            printf("Ticket cancellation successful.\n");
            found = 1;
            break;
        }
    }
    if (!found) printf("Flight not found.\n");
}

void displayByDestination(struct Flight flights[], int count)
 {
    char destination[50];
    printf("Enter Destination: ");
    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);
        }
    }
}
```