Assignment

```
// 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.
{
   int *arr,initial_size,new_size,i,choice;
   printf("Enter the initial size of array:");
   scanf("%d",&initial_size);
   arr=(int *)malloc(initial_size * sizeof(int));
   printf("Enter elements:");
   for(i=0;i<initial_size;i++)
   {
      scanf("%d",&arr[i]);
   }
   while(1)
   {
      printf("Choose an operation:\n");
      printf("1.Increase size \n");
      printf("2.Decrease size \n");
      printf("3.Exit \n");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         printf("Enter new size:");
         scanf("%d",&new_size);
```

```
        arr=(int *)realloc(arr,new_size * sizeof(int));
        if (new_size > initial_size) {
            printf("Enter %d more elements:\n", new_size - initial_size);
            for (i = initial_size; i < new_size; i++) {
                scanf("%d", &arr[i]);
            }
        }
        initial_size = new_size;
        break;

        case 2:
        printf("Enter the new size:");
        scanf("%d",&new_size);
        arr = (int *)realloc(arr, new_size * sizeof(int));
        initial_size=new_size;
        break;

        case 3:
        free(arr);
        printf("Exiting the program.\n");
        return 0;

        default:
        printf("Invalid choice");
    }
    printf("Array after resizing");
    for (i = 0; i < initial_size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
    return 0;
}
```

Output:

```
Enter the initial size of array:5
Enter elements:1 2 3 4 5
Choose an operation:
1.Increase size
2.Decrease size
3.Exit
1
Enter new size:10
Enter 5 more elements:
6 7 8 9 10
Array after resizing1 2 3 4 5 6 7 8 9 10
Choose an operation:
1.Increase size
2.Decrease size
3.Exit
2
Enter the new size:5
Array after resizing1 2 3 4 5
Choose an operation:
1.Increase size
2.Decrease size
3.Exit
3
Exiting the program.
```

```
// Problem 2: String Concatenation Using Dynamic Memory
// Objective: Create a program that concatenates two strings using dynamic memory allocation.
// Description:
// Accept two strings from the user.
// Use malloc to allocate memory for the first string.
// Use realloc to resize the memory to accommodate the concatenated string.
// Concatenate the strings and print the result.
// Free the allocated memory.
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
  char *str1,*str2,*res;
  int len1,len2;
  str1=(char *)malloc(100 * sizeof(char));
  str2=(char *)malloc(100 * sizeof(char));
  printf("Enter first string:");
  scanf("%s",str1);
  printf("Enter second string:");
  scanf("%s",str2);
  len1=strlen(str1);
  len2=strlen(str2);
  res=(char *)realloc(str1,(len1+len2+1) * sizeof(char));
  strcat(res,str2);
  printf("Concatenated string:%s\n",res);
  free(res);
  free(str2);
  return 0;
}
```

Output:

Enter first string:Sanjana

Enter second string: Haridas

Concatenated string:SanjanaHaridas

// Problem 3: Sparse Matrix Representation

// Objective: Represent a sparse matrix using dynamic memory allocation.

// Description:

// Accept a matrix of size m×nm \times nm×n from the user.

// Store only the non-zero elements in a dynamically allocated array of structures (with fields for row, column, and value).

// Print the sparse matrix representation.

// Free the allocated memory at the end.

```
 #include<stdio.h>
#include<stdlib.h>
struct SparseMatrix{
   int row;
   int col;
   int val;
};
int main()
{
   int m,n,non_zero=0,k=0;
   printf("Enter the number of rows:");
   scanf("%d",&m);
   printf("Enter number of col:");
   scanf("%d",&n);
   int matrix[m][n];
   printf("Enter matrix elements:\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)
        {
            non_zero++;
        }
    }
}
struct SparseMatrix *sparse=(struct SparseMatrix *)malloc(non_zero * sizeof(struct
SparseMatrix));
for(int i=0;i<m;i++)
{
    for(int j=0;j<n;j++)
    {
        if(matrix[i][j]!=0)
        {
            sparse[k].row=i;
            sparse[k].col=j;
            sparse[k].val=matrix[i][j];
            k++;
        }
    }
}
printf("Sparse matrix representation:");
for(int i=0;i<non_zero;i++)
{
    printf("\n %d %d %d \n",sparse[i].row,sparse[i].col,sparse[i].val);
}
free(sparse);
return 0;
```

}

Output:

Enter the number of rows:2

Enter number of col:2

Enter matrix elements:

1

2

3

4

Sparse matrix representation:

0 0 1

0 1 2

1 0 3

1 1 4

// Problem 4: Dynamic Linked List Implementation

// Objective: Implement a linked list using dynamic memory allocation.

// Description:

// Define a struct for linked list nodes. Each node should store an integer and a pointer to the next node.

// 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.

// Use malloc to allocate memory for each new node and free to deallocate memory for deleted nodes.

```
// Problem 5: Dynamic 2D Array Allocation
// Objective: Write a program to dynamically allocate a 2D array.
// Description:
// Accept the number of rows and columns from the user.
// Use malloc (or calloc) to allocate memory for the rows and columns dynamically.
// Allow the user to input values into the 2D array.
// Print the array in matrix format.
// Free all allocated memory at the end.
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int row,col,**matrix;
   printf("Enter the rows:");
   scanf("%d",&row);
   printf("Enter the columns:");
   scanf("%d",&col);
   matrix=(int **)malloc(row * sizeof(int *));
   for(int i=0;i<row;i++)
   {
     matrix[i]=(int *)malloc(col * sizeof(int)) ;
   }
   printf("Enter the elements of the matrix:\n");
   for(int i=0;i<row;i++)
   {
      for(int j=0;j<col;j++)
      {
         scanf("%d",&matrix[i][j]);
      }
   }
```

```
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < row; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

Output:

```
Enter the rows:3
Enter the columns:3
Enter the elements of the matrix:
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
```

Problem 1: Student Record Management System Objective Create a program to manage student records using structures. Requirements 1. Define a Student structure with the following fields: o 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<string.h>

struct student{


   char name[50];


   int rollNumber;


   float marks;


};


int main()

{

  struct  student students[100];

  int count=0,choice;

  while(1){

     printf("1.Add student\n");

     printf("2.Display all students \n");

     printf("3.Find students by roll number \n");

     printf("4.Calculate average marks \n");

     printf("5.Exit \n");
```

```
printf("Enter your choice:");
scanf("%d",&choice);
if(choice==1)
{
   if (count >= 100) {
       printf("Cannot add more students. Maximum limit reached.\n");
   }
   else
   {
       printf("Enter name: ");
       scanf(" %[^\n]", students[count].name);
       printf("Enter roll number: ");
       scanf("%d", &students[count].rollNumber);
       printf("Enter marks: ");
       scanf("%f", &students[count].marks);
       count++;
       printf("Student added successfully!\n");

    }
}
else if(choice==2)
{
   if(count==0)
   {
      printf("No records found.\n");
   }
   else
   {
      for(int i=0;i<count;i++)
      {
```

```
            printf("Name:%s,Roll
Number:%d,Marks:%.2f\n",students[i].name,students[i].rollNumber,students[i].marks);
        }
    }
}
else if(choice==3)
{
    if(count==0)
    {
        printf("No records found.\n");
    }
    else
    {
        int rollNumber,found=0;
        printf("Enter the roll number:");
        scanf("%d",&rollNumber);
        for(int i=0;i<count;i++)
        {
            if(students[i].rollNumber==rollNumber)
            {
                printf("Student found: Name:%s, Marks:%.2f\n",students[i].name,students[i].marks);
                found=1;
                break;
            }
        }
        if(!found)
        {
            printf("No student found with roll number %d.\n",rollNumber);
        }
    }
}
```

```
    else if(choice==4)
    {
       if(count==0)
       {
          printf("No student records found \n");
       }
       else
       {
          float total=0;
          for(int i=0;i<count;i++)
          {
             total+=students[i].marks;
          }
          printf("Average marks:%.2f\n",total/count);
       }
    }
    else if(choice==5)
    {
       printf("Exit the program.\n");
       break;
    }
    else
    {
       printf("Invalid choice");
    }
  }
  return 0;
}
```

Output:

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:1

Enter name: anu

Enter roll number: 001

Enter marks: 89

Student added successfully!

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:1

Enter name: arun

Enter roll number: 002

Enter marks: 69

Student added successfully!

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:2

Name:anu,Roll Number:1,Marks:89.00

Name:arun,Roll Number:2,Marks:69.00

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:3

Enter the roll number:002

Student found: Name:arun, Marks:69.00

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:4

Average marks:79.00

1.Add student

2.Display all students

3.Find students by roll number

4.Calculate average marks

5.Exit

Enter your choice:5

Exit the program.

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>
#include<stdlib.h>
struct Employee{
   int emp_id;
   char name[50];
   float salary;
};
int main()
{
   struct Employee employees[100];
   int count=0,choice,id;
   float new_salary;
   while(1){


      printf("1.Add employee \n");
      printf("2.Update employee salary by id \n");
      printf("3.Display all employees \n");
      printf("4.Employee with highest salary \n");
      printf("5.Exit \n");
      printf("Enter your choice: \n");
      scanf("%d",&choice);
      switch (choice)
      {
         case 1:
         printf("Enter employee id:");
         scanf("%d",&employees[count].emp_id);
         printf("Enter name:");
         scanf("%s",employees[count].name);
```

```
printf("Enter salary:");
scanf("%f",&employees[count].salary);
count++;
printf("Employee added successfully \n");
break;


case 2:
printf("Enter employee id to update salary: ");
scanf("%d",&id);
int found=0;
for(int i=0;i<count;i++)
{
   if(employees[i].emp_id==id)
   {
      printf("Enter new salary:");
      scanf("%f",&new_salary);
      employees[i].salary=new_salary;
      printf("Salary updated successfully \n");
      found=1;
      break;
   }
}
if(!found){
   printf("Employee id not found");
}
break;


case 3:
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);
        }
        break;

        case 4:
        if(count==0)
        {
            printf("No employees");
        }
        else
        {
            int max_index=0;
            for(int i=0;i<count;i++)
            {
                if(employees[i].salary > employees[max_index].salary)
                {
                    max_index=i;
                }
            }
            printf("Employee with Highest Salary:\n");
            printf("ID: %d, Name: %s, Salary: %.2f\n", employees[max_index].emp_id,
employees[max_index].name, employees[max_index].salary);
        }
        break;

        case 5:
        printf("Exit program");
        return 0;
```

```
            default:
            printf("Invalid choice \n");
        }
    }
    return 0;
}
```

Output:

```
1.Add employee
2.Update employee salary by id
3.Display all employees
4.Employee with highest salary
5.Exit
Enter your choice:
1
Enter employee id:011
Enter name:ahan
Enter salary:20000
Employee added successfully
1.Add employee
2.Update employee salary by id
3.Display all employees
4.Employee with highest salary
5.Exit
Enter your choice:
1
Enter employee id:025
Enter name:anu
Enter salary:28000
Employee added successfully
1.Add employee
2.Update employee salary by id
```

3.Display all employees

4.Employee with highest salary

5.Exit

Enter your choice:

1

Enter employee id:035

Enter name:abhishek

Enter salary:30000

Employee added successfully

1.Add employee

2.Update employee salary by id

3.Display all employees

4.Employee with highest salary

5.Exit

Enter your choice:

2

Enter employee id to update salary: 011

Enter new salary:27000

Salary updated successfully

1.Add employee

2.Update employee salary by id

3.Display all employees

4.Employee with highest salary

5.Exit

Enter your choice:

3

Employee details:

ID:11, Name: ahan, Salary: 27000.00

ID:25, Name: anu, Salary: 28000.00

ID:35, Name: abhishek, Salary: 30000.00

1.Add employee

2.Update employee salary by id

3.Display all employees

4.Employee with highest salary

5.Exit

Enter your choice:

4

Employee with Highest Salary:

ID: 35, Name: abhishek, Salary: 30000.00

1.Add employee

2.Update employee salary by id

3.Display all employees

4.Employee with highest salary

5.Exit

Enter your choice:

5

Exit program

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>
#include<stdlib.h>
struct Book{
   int book_id;
   char title[100];
   char author[50];
   int copies;
};
int main()
{
   struct Book library[100];
   int count=0,choice,id;
   char search[100];
   while(1){


      printf("1.Add book \n");
      printf("2.Issue book \n");
      printf("3.Return book \n");
      printf("4.Search book by title or author name \n");
      printf("5.Exit \n");
      printf("Enter your choice: \n");
      scanf("%d",&choice);
      switch (choice)
      {
         case 1:
         printf("Enter book id:");
```

```
scanf("%d",&library[count].book_id);
printf("Enter title:");
scanf("%s",library[count].title);
printf("Enter author:");
scanf("%s",library[count].author);
printf("Enter number of copies:");
scanf("%d",&library[count].copies);
count++;
printf("Book added successfully \n");
break;



case 2:
printf("Enter book id to issue: ");
scanf("%d",&id);
int found=0;
for(int i=0;i<count;i++)
{
   if(library[i].book_id==id)
   {
      if(library[i].copies>0)
      {
         library[i].copies--;
         printf("Book issued successfully");
      }
      else
      {
         printf("No copies are available");
      }
      found=1;
      break;
```

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

case 3:
printf("Enter book id to return: \n");
scanf("%d",&id);
found=0;
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 with ID %d not found.\n", id);
}
break;



case 4:
printf("Enter Title or Author to search: ");
```

```
            scanf(" %s", search);
            found = 0;
            for (int i = 0; i < count; i++)
            {
               if (strstr(library[i].title, search) || strstr(library[i].author, search))
               {
                  printf("Book Found - ID: %d, Title: %s, Author: %s, Copies: %d\n",
                  library[i].book_id, library[i].title, library[i].author, library[i].copies);
                  found = 1;
               }
            }
            if (!found)
            {
               printf("No book found with Title/Author %s.\n", search);
            }
            break;

            case 5:
               printf("Exiting program.\n");
               return 0;

            default:
            printf("Invalid choice.\n");
         }
      }
      return 0;
}
```

Output:

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

1

Enter book id: 101

Enter title: The Great Gatsby

Enter author: Scott

Enter number of copies: 5

Book added successfully

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

2

Enter book id to issue: 101

Book issued successfully

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

2

Enter book id to issue: 101

No copies are available

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

3

Enter book id to return: 101

Book returned successfully!

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

4

Enter Title or Author to search: Gatsby

Book Found - ID: 101, Title: The Great Gatsby, Author: Scott, Copies: 5

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

4

Enter Title or Author to search: Harry Potter

No book found with Title/Author Harry Potter.

1.Add book

2.Issue book

3.Return book

4.Search book by title or author name

5.Exit

Enter your choice:

5

Exiting program.

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

int main() {
   struct Player players[100];
```

```
int n = 0, choice;
int total = 0;

while (1) {
   printf("1. Add Player\n");
   printf("2. Display All Players\n");
   printf("3. Display Player with Highest Average\n");
   printf("4. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         printf("Enter number of players to add: ");
         scanf("%d", &n);
         for (int i = total; i < total + n; i++) {
            printf("Enter player %d details:\n", i + 1);
            printf("Name: ");
            scanf(" %[^\n]", players[i].name);
            printf("Matches played: ");
            scanf("%d", &players[i].matches);
            printf("Total runs: ");
            scanf("%d", &players[i].runs);

            if (players[i].matches != 0) {
               players[i].average = (float)players[i].runs / players[i].matches;
            } else {
               players[i].average = 0.0;
            }
         }
         total += n;
```

```
            break;

        case 2:
            printf("Displaying all the players \n");
            for (int i = 0; i < total; i++) {
                printf("Player: %s, Matches: %d, Runs: %d, Average: %.2f\n",players[i].name,
players[i].matches, players[i].runs, players[i].average);
            }
            break;

        case 3:
            if (total > 0) {
                int highest_index = 0;
                for (int i = 1; i < total; i++) {
                    if (players[i].average > players[highest_index].average) {
                        highest_index = i;
                    }
                }
                printf("Player with highest average: %s, Average: %.2f\n",
                    players[highest_index].name, players[highest_index].average);
            } else {
                printf("No players available \n");
            }
            break;

        case 4:
            printf("Exiting program.\n");
            return 0;

        default:
            printf("Invalid choice\n");
```

```
        }
    }

    return 0;
}
```

Output:

```
1. Add Player
2. Display All Players
3. Display Player with Highest Average
4. Exit
Enter your choice:
1
Enter number of players to add: 2
Enter player 1 details:
Name: Virat Kohli
Matches played: 50
Total runs: 2500
Enter player 2 details:
Name: Rohit Sharma
Matches played: 40
Total runs: 1600

1. Add Player
2. Display All Players
3. Display Player with Highest Average
4. Exit
Enter your choice:
2
Displaying all the players
Player: Virat Kohli, Matches: 50, Runs: 2500, Average: 50.00
Player: Rohit Sharma, Matches: 40, Runs: 1600, Average: 40.00
```

1. Add Player

2. Display All Players

3. Display Player with Highest Average

4. Exit

Enter your choice:

3

Player with highest average: Virat Kohli, Average: 50.00

1. Add Player

2. Display All Players

3. Display Player with Highest Average

4. Exit

Enter your choice:

4

Exiting program.

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;
};

int main() {
    struct Student students[100];
    int n = 0, choice;
    int total = 0;

    while (1) {
        printf("1. Add Student\n");
        printf("2. Display All Students\n");
        printf("3. Assign grades\n");
        printf("4.Exit \n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter number of students to add: ");
                scanf("%d", &n);
                for (int i = total; i < total + n; i++) {
                    printf("Enter student %d details:\n", i + 1);
                    printf("Roll number: ");
                    scanf(" %d", &students[i].roll_no);
                    printf("Name: ");
                    scanf("%s", students[i].name);
```

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

case 2:
    printf("Displaying all the students \n");
    for (int i = 0; i < total; i++) {
        printf("Roll No: %d, Name: %s, Marks: \n",students[i].roll_no, students[i].name);
        for(int j=0;j<5;j++)
        {
            printf("%.2f",students[i].marks[j]);
        }
        printf("\n");
    }
    break;

case 3:
    printf("\nAssigning Grades to Students:\n");
    for (int i = 0; i < total; i++) {
        float marks = 0;
        for (int j = 0; j < 5; j++) {
            marks += students[i].marks[j];
        }
        float avg = marks / 5;

        if (avg >= 90) {
```

```
                students[i].grade = 'A';
            } else if (avg >= 75) {
                students[i].grade = 'B';
            } else if (avg >= 60) {
                students[i].grade = 'C';
            } else if (avg >= 50) {
                students[i].grade = 'D';
            } else {
                students[i].grade = 'F';
            }
            printf("Student %s, Grade: %c\n", students[i].name, students[i].grade);
        }
        break;

    case 4:
        printf("Exiting program.\n");
        return 0;

    default:
        printf("Invalid choice\n");
  }
 }

 return 0;
}
```

Output:

1. Add Student
2. Display All Students
3. Assign grades
4. Exit

Enter your choice:

1

Enter number of students to add: 2

Enter student 1 details:

Roll number: 101

Name: John

Enter marks in 5 subjects: 85 90 75 88 92

Enter student 2 details:

Roll number: 102

Name: Alice

Enter marks in 5 subjects: 78 72 80 85 95

1. Add Student

2. Display All Students

3. Assign grades

4. Exit

Enter your choice:

2

Displaying all the students

Roll No: 101, Name: John, Marks:

85.00 90.00 75.00 88.00 92.00

Roll No: 102, Name: Alice, Marks:

78.00 72.00 80.00 85.00 95.00

1. Add Student

2. Display All Students

3. Assign grades

4. Exit

Enter your choice:

3

Assigning Grades to Students:

Student John, Grade: A

Student Alice, Grade: B

1. Add Student

2. Display All Students

3. Assign grades

4. Exit

Enter your choice:

4

Exiting program.

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>

struct Flight {
   char flight_number[10];
   char destination[50];
```

```
    int available_seats;
};

int main() {
    struct Flight flights[100];
    int count = 0, choice, seats, found;
    char destination[50];

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

        switch (choice) {
            case 1:
                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++;
                printf("Flight added successfully!\n");
                break;

            case 2:
                printf("Enter Flight Number to book: ");
```

```
    scanf(" %[^\n]", flights[i].flight_number);
    printf("Enter number of seats to book: ");
    scanf("%d", &seats);
    found = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(flights[i].flight_number, flights[i].flight_number) == 0) {
            if (flights[i].available_seats >= seats) {
                flights[i].available_seats -= seats;
                printf("Booking successful! Remaining seats: %d\n", flights[i].available_seats);
            } else {
                printf("Not enough seats available.\n");
            }
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Flight not found.\n");
    }
    break;

case 3:
    printf("Enter destination to view flight details: ");
    scanf(" %[^\n]", destination);
    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:
    printf("Enter Flight Number to cancel tickets: ");
    scanf(" %[^\n]", flights[count].flight_number);
    printf("Enter number of seats to cancel: ");
    scanf("%d", &seats);
    found = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(flights[i].flight_number, flights[count].flight_number) == 0) {
            flights[i].available_seats += seats;
            printf("Ticket cancellation successful! Remaining seats: %d\n",flights[i].available_seats);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Flight not found.\n");
    }
    break;

case 5:
    printf("Exiting program.\n");
    return 0;

default:
    printf("Invalid choice.\n");
```

```
        }
    }
}
```

Output:

```
1. Add Flight
2. Book Ticket
3. Display Flight Details by Destination
4. Cancel Ticket
5. Exit
Enter your choice: 1
Enter Flight Number: AA123
Enter Destination: New York
Enter Available Seats: 100
Flight added successfully!

1. Add Flight
2. Book Ticket
3. Display Flight Details by Destination
4. Cancel Ticket
5. Exit
Enter your choice: 1
Enter Flight Number: BB456
Enter Destination: Los Angeles
Enter Available Seats: 150
Flight added successfully!

1. Add Flight
2. Book Ticket
3. Display Flight Details by Destination
4. Cancel Ticket
5. Exit
```

Enter your choice: 3

Enter destination to view flight details: Los Angeles

Flight Number: BB456, Destination: Los Angeles, Available Seats: 150

1. Add Flight

2. Book Ticket

3. Display Flight Details by Destination

4. Cancel Ticket

5. Exit

Enter your choice: 2

Enter Flight Number to book: AA123

Enter number of seats to book: 10

Booking successful! Remaining seats: 90

1. Add Flight

2. Book Ticket

3. Display Flight Details by Destination

4. Cancel Ticket

5. Exit

Enter your choice: 4

Enter Flight Number to cancel tickets: AA123

Enter number of seats to cancel: 5

Ticket cancellation successful! Remaining seats: 95

1. Add Flight

2. Book Ticket

3. Display Flight Details by Destination

4. Cancel Ticket

5. Exit

Enter your choice: 5

Exiting program.