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.

```c
#include<stdio.h>

#include<stdlib.h>

void printarray(int *array,int size);

int main(){

    int *array=NULL;

    int size,new;

    printf("\nEnter size of array::");

    scanf("%d",&size);

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

    if(array==NULL){

        printf("\nAllocation Error");

    }

    else{

        printf("\nEnter %d elements into the array:",size);

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

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

        }

    }
```

```
    printarray(array,size);
    printf("\nEnter new reallocation size::");
    scanf("%d",&new);
    array=(int*)realloc(array,new*sizeof(int));
    if(new>size){
        for(int i=size;i<new;i++){
            array[i]=0;
        }
    }

    printf("\nEnter %d new elements into array:",new-size);
    for(int i=size;i<new;i++){
        scanf("%d",&array[i]);
    }
    size=new;
    printf("\nNew Array is::");
    printarray(array,size);
    free(array);
    return 0;


}
void printarray(int *array,int size){
    printf("\nArray is::");
    for(int i=0;i<size;i++){
```

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

    }

    printf("\n");

}
```

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, *str2 = NULL, *result = NULL;

    int len1, len2;

    printf("Enter the first string: ");

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

    if (str1 == NULL) {

        printf("Allocation Error!\n");

        return 1;

    }
```

```
    scanf("%s", str1);
    len1 = strlen(str1);



    printf("Enter the second string: ");
    str2 = (char *)malloc(100 * sizeof(char));
    if (str2 == NULL) {
        printf("Allocation Error!\n");
        free(str1);
        return 1;
    }
    scanf("%s", str2);
    len2 = strlen(str2);



    result = (char *)realloc(str1, (len1 + len2 + 1) * sizeof(char));
    if (result == NULL) {
        printf("Allocation Error!\n");
        free(str1);
        free(str2);
        return 1;
    }



    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.

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.

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.

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

int main() {
    int rows, cols;


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


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

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

```
        array[i] = (int*)malloc(cols * sizeof(int));
        if (array[i] == NULL) {
            printf("Error!\n");
            return 1;
        }
    }


    printf("Enter values for the 2D array:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("Enter value for 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.Struct Student.

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

struct student {
    char name[50];
```

```
    int roll;
    float mark;
};

int main() {
    int op, i = 0, r;
    struct student std[20];
    int count = 0;
    float sum = 0, avg;

    while(1) {

        printf("\n1.Add Student\n2.Display all students\n3.Find student
by roll number\n4.Calculate avg mark\n5.Exit");
        printf("\nEnter choice: ");
        scanf("%d", &op);

        switch(op) {
            case 1:
                if(count < 20) {
                    printf("\nEnter name: ");
                    scanf("%s", std[i].name);
                    printf("\nEnter roll no: ");
                    scanf("%d", &std[i].roll);
                    printf("\nEnter mark: ");
                    scanf("%f", &std[i].mark);
                    printf("\nStudent Added Successfully\n");

                    count++;
                    i++;
                } else {
                    printf("\nList Full\n");
                }
                break;

            case 2:
                if(count == 0) {
                    printf("\nNo students to display.\n");
                } else {
                    printf("\nDisplaying all students:\n");
                    for(i = 0; i < count; i++) {
                        printf("\nName: %s, Roll No: %d, Marks: %.2f",
std[i].name, std[i].roll, std[i].mark);
```

```
                    }
                }
                break;

            case 3:
                printf("\nEnter roll no to check: ");
                scanf("%d", &r);
                int found = 0;
                for(i = 0; i < count; i++) {
                    if(std[i].roll == r) {
                        printf("\nFound: %s, Roll No: %d, Marks: %.2f",
std[i].name, std[i].roll, std[i].mark);
                        found = 1;
                        break;
                    }
                }
                if (!found) {
                    printf("\nNOT Found");
                }
                break;

            case 4:
                if(count == 0) {
                    printf("\nNo students to calculate average
marks.\n");
                } else {
                    sum = 0;
                    for(i = 0; i < count; i++) {
                        sum += std[i].mark;
                    }
                    avg = sum / count;
                    printf("\nAverage mark is: %.2f", avg);
                }
                break;

            case 5:
                printf("\nExiting program...\n");
                exit(0);

            default:
                printf("\nInvalid choice. Try again.\n");
        }
    }
```

```
    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.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



struct Employee {

    int emp_id;

    char name[50];

    float salary;

};



int main() {

    struct Employee emp[50];

    int count = 0, choice, id, i;
```

```c
    float max_salary;
    int max_index;


    while (1) {
        printf("\n\n1. Add Employee");
        printf("\n2. Update Employee Salary by ID");
        printf("\n3. Display All Employees");
        printf("\n4. Find Employee with Highest Salary");
        printf("\n5. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);


        switch (choice) {
            case 1:
                if (count < 50) {
                    printf("\nEnter Employee ID: ");
                    scanf("%d", &emp[count].emp_id);
                    printf("Enter Employee Name: ");
                    scanf("%s", emp[count].name);
                    printf("Enter Employee Salary: ");
                    scanf("%f", &emp[count].salary);
                    printf("\nEmployee Added Successfully!\n");
                    count++;
                } else {
                    printf("\nEmployee list is full!\n");
```

```
                }
                break;

            case 2:
                if (count == 0) {
                    printf("\nNo employees available to update.\n");
                } else {
                    printf("\nEnter Employee ID to update salary: ");
                    scanf("%d", &id);
                    int found = 0;
                    for (i = 0; i < count; i++) {
                        if (emp[i].emp_id == id) {
                            printf("Current Salary: %.2f\n",
emp[i].salary);
                            printf("Enter New Salary: ");
                            scanf("%f", &emp[i].salary);
                            printf("\nSalary Updated Successfully!\n");
                            found = 1;
                            break;
                        }
                    }
                    if (!found) {
                        printf("\nEmployee with ID %d not found!\n",
id);
                    }
                }
```

```
                break;


            case 3:
                if (count == 0) {
                    printf("\nNo employees to display.\n");
                } else {
                    printf("\nAll Employee Details:\n");
                    for (i = 0; i < count; i++) {
                        printf("\nID: %d, Name: %s, Salary: %.2f",
emp[i].emp_id, emp[i].name, emp[i].salary);
                    }
                }
                break;


            case 4:
                if (count == 0) {
                    printf("\nNo employees to evaluate.\n");
                } else {
                    max_salary = emp[0].salary;
                    max_index = 0;
                    for (i = 1; i < count; i++) {
                        if (emp[i].salary > max_salary) {
                            max_salary = emp[i].salary;
                            max_index = i;
                        }
```

```
                    }

                    printf("\nEmployee with the Highest Salary:");

                    printf("\nID: %d, Name: %s, Salary: %.2f",
emp[max_index].emp_id, emp[max_index].name, emp[max_index].salary);

                }

                break;



            case 5:

                printf("\nExiting program...\n");

                exit(0);



            default:

                printf("\nInvalid choice. Please try again.\n");

        }

    }



    return 0;

}
```

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.

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

```c
#include <stdlib.h>

#include <string.h>



struct Book {

    int book_id;

    char title[100];

    char author[50];

    int copies;

};



int main() {

    struct Book library[50];

    int count = 0, choice, id, i;

    char search_query[100];



    while (1) {

        printf("\n\n1. Add Book");
```

```
        printf("\n2. Issue Book");
        printf("\n3. Return Book");
        printf("\n4. Search Book by Title or Author");
        printf("\n5. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (count < 50) {
                    printf("\nEnter 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);
                    printf("\nBook Added Successfully!\n");
                    count++;
                } else {
                    printf("\nLibrary is full! Cannot add more
books.\n");
                }
                break;
```

```
            case 2:
                if (count == 0) {
                    printf("\nNo books available in the library.\n");
                } else {
                    printf("\nEnter Book ID to issue: ");
                    scanf("%d", &id);
                    int found = 0;
                    for (i = 0; i < count; i++) {
                        if (library[i].book_id == id) {
                            if (library[i].copies > 0) {
                                library[i].copies--;
                                printf("\nBook issued successfully!
Remaining copies: %d\n", library[i].copies);
                            } else {
                                printf("\nNo copies available for this
book.\n");
                            }
                            found = 1;
                            break;
                        }
                    }
                    if (!found) {
                        printf("\nBook with ID %d not found!\n", id);
                    }
                }
```

```
                break;


            case 3:
                if (count == 0) {
                    printf("\nNo books available in the library.\n");
                } else {
                    printf("\nEnter Book ID to return: ");
                    scanf("%d", &id);
                    int found = 0;
                    for (i = 0; i < count; i++) {
                        if (library[i].book_id == id) {
                            library[i].copies++;
                            printf("\nBook returned successfully! Total
copies: %d\n", library[i].copies);
                            found = 1;
                            break;
                        }
                    }
                    if (!found) {
                        printf("\nBook with ID %d not found!\n", id);
                    }
                }
                break;


            case 4:
```

```
                if (count == 0) {
                    printf("\nNo books available in the library.\n");
                } else {
                    printf("\nEnter title or author to search: ");
                    scanf(" %[^\n]", search_query);
                    int found = 0;
                    for (i = 0; i < count; i++) {
                        if (strstr(library[i].title, search_query) ||
strstr(library[i].author, search_query)) {
                            printf("\nBook Found: ID: %d, Title: %s,
Author: %s, Copies: %d",
                                   library[i].book_id,
library[i].title, library[i].author, library[i].copies);
                            found = 1;
                        }
                    }
                    if (!found) {
                        printf("\nNo books found matching the query
\"%s\".\n", search_query);
                    }
                }
                break;

            case 5:
                printf("\nExiting program...\n");
                exit(0);
```

```
            default:
                printf("\nInvalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

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 <stdlib.h>

#include <string.h>
```

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

int main() {
    struct Player players[50];
    int n, i;
    int max_index = 0;
    float max_average = 0;

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

    for (i = 0; i < n; i++) {
        printf("\nEnter details for player %d:\n", i + 1);
        printf("Enter name: ");
        scanf(" %[^\n]", players[i].name);
        printf("Enter number of matches played: ");
        scanf("%d", &players[i].matches);
        printf("Enter total runs scored: ");
        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;
        }

        if (players[i].average > max_average) {
            max_average = players[i].average;
            max_index = i;
        }
    }

    printf("\nPlayer Statistics:\n");
    for (i = 0; i < n; i++) {
        printf("\nName: %s, Matches: %d, Runs: %d, Batting Average:
%.2f",
               players[i].name, players[i].matches, players[i].runs,
players[i].average);
    }

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

```
    return 0;

}
```

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>

#include <stdlib.h>



struct Student {

    int roll_no;

    char name[50];

    float marks[5];

    char grade;

};

char calculate_grade(float average);
```

```c
int main() {
    int n, i, j;
    struct Student students[50];

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

    for (i = 0; i < n; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Enter roll number: ");
        scanf("%d", &students[i].roll_no);
        printf("Enter name: ");
        scanf(" %[^\n]", students[i].name);

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

        float average = total / 5.0;
```

```
        students[i].grade = calculate_grade(average);
    }

    printf("\nStudent Details:\n");
    for (i = 0; i < n; i++) {
        printf("\nRoll No: %d, Name: %s, Marks: ", students[i].roll_no,
students[i].name);
        for (j = 0; j < 5; j++) {
            printf("%.2f ", students[i].marks[j]);
        }
        printf(", Grade: %c", students[i].grade);
    }

    return 0;
}
char calculate_grade(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';

    }

}
```

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

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

int main() {
    struct Flight flights[50];
    int n, choice, i, seats, found;
    char dest[50];

    printf("Enter the number of flights: ");
```

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


    for (i = 0; i < n; i++) {
        printf("\nEnter details for flight %d:\n", i + 1);
        printf("Enter flight number: ");
        scanf("%s", flights[i].flight_number);
        printf("Enter destination: ");
        scanf(" %[^\n]", flights[i].destination);
        printf("Enter available seats: ");
        scanf("%d", &flights[i].available_seats);
    }

    while (1) {
        printf("\n1. Book Ticket\n2. Cancel Ticket\n3. Display Flights
by Destination\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\nEnter destination to book ticket: ");
                scanf(" %[^\n]", dest);

                found = 0;
                for (i = 0; i < n; i++) {
                    if (strcmp(flights[i].destination, dest) == 0) {
                        found = 1;
                        printf("Enter number of seats to book: ");
                        scanf("%d", &seats);

                        if (seats <= flights[i].available_seats) {
                            flights[i].available_seats -= seats;
                            printf("%d seats booked for flight %s to
%s. Remaining seats: %d\n",
                                   seats, flights[i].flight_number,
flights[i].destination, flights[i].available_seats);
                        } else {
                            printf("Not enough seats available.\n");
                        }
                        break;
                    }
                }
```

```
                if (!found) {
                    printf("No flight available to %s.\n", dest);
                }
                break;

            case 2:
                printf("\nEnter destination to cancel ticket: ");
                scanf(" %[^\n]", dest);

                found = 0;
                for (i = 0; i < n; i++) {
                    if (strcmp(flights[i].destination, dest) == 0) {
                        found = 1;
                        printf("Enter number of seats to cancel: ");
                        scanf("%d", &seats);

                        flights[i].available_seats += seats;
                        printf("%d seats canceled for flight %s to %s.
Available seats: %d\n",
                               seats, flights[i].flight_number,
flights[i].destination, flights[i].available_seats);
                        break;
                    }
                }
                if (!found) {
                    printf("No flight available to %s.\n", dest);
                }
                break;

            case 3:
                printf("\nEnter destination to display flights: ");
                scanf(" %[^\n]", dest);

                found = 0;
                for (i = 0; i < n; i++) {
                    if (strcmp(flights[i].destination, dest) == 0) {
                        found = 1;
                        printf("\nFlight Number: %s, Destination: %s,
Available Seats: %d\n",
                               flights[i].flight_number,
flights[i].destination, flights[i].available_seats);
                    }
                }
```

```
                if (!found) {
                    printf("No flights available to %s.\n", dest);
                }
                break;

            case 4:
                printf("\nExiting program...\n");
                exit(0);

            default:
                printf("\nInvalid choice. Please try again.\n");
        }
    }

    return 0;
}
```