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>
int main() {
int size, size2;
char choice;
int *arr;
printf("Enter the size: ");
scanf("%d", &size);
// Allocate memory
arr = (int *)malloc(size * sizeof(int));
if (arr == NULL) {
printf("Allocation failed\n");
return 1;
}
// Enter elements
printf("Enter %d elements:\n", size);
for (int i = 0; i < size; i++) {
scanf("%d", &arr[i]); // Fixed syntax: removed `&` from `&arr[i]`
}
// Print elements
printf("Array elements: ");
for (int i = 0; i < size; i++) {
printf("%d ", arr[i]); // Fixed typo: `printff` -> `printf`
}
printf("\n");
while (1) {
printf("Do you want to resize? (yes/no): ");
scanf(" %c", &choice); // Added space before `%c` to handle newline character
if (choice == 'n') { // Fixed comparison to check for 'no' (or 'n')
break;
}
printf("Enter the new size of the array: ");
scanf("%d", &size2);
// Resize array
arr = (int *)realloc(arr, size2 * sizeof(int));
if (arr == NULL) {
printf("Reallocation failed\n");
return 1;
}
// If increasing size, prompt for new elements
if (size2 > size) {
printf("Enter %d new elements:\n", size2 - size);
for (int i = size; i < size2; i++) {
scanf("%d", &arr[i]);
}
}
size = size2; // Update the current size
// Print updated array
printf("Updated array elements: ");
for (int i = 0; i < size; i++) {
printf("%d ", arr[i]);
}
printf("\n");
}
// Free memory
free(arr);
printf("Memory freed and program exited.\n");
return 0;
}
```

Problem 2: String Concatenation Using Dynamic Memory
Objective: Create a program that concatenates two strings using dynamic memory allocation.
Description:
1. Accept two strings from the user.
2. Use malloc to allocate memory for the first string.
3. Use realloc to resize the memory to accommodate the concatenated string.
4. Concatenate the strings and print the result.
5. Free the allocated memory.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
char str1 = (char)malloc(100 * sizeof(char));
if (str1 == NULL) {
printf("Memory allocation failed\n");
return 1;
}
char str2[100];
printf("Enter the first string: ");
scanf("%[^\n]", str1);
getchar();
printf("Enter the second string: ");
scanf("%[^\n]", str2);
str1 = (char*)realloc(str1,(strlen(str2)+1));
strcat(str1, str2);
printf("Concatenated string: %s\n", str1);
free(str1);
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.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
int row;
int col;
int val;
}s_matrix;
int main()
{
int m, n, count=0;
printf("Enter the number of rows and columns of the matrix: ");
scanf("%d %d", &m, &n);
int** matrix = (int**)malloc(m * sizeof(int *));
for (int i = 0; i < m; i++)
{
matrix[i] = (int*)malloc(n * sizeof(int));
}
printf("Enter the elements of the matrix:\n");
for (int i = 0; i < m; i++)
{
for (int j = 0; j < n; j++)
{
scanf("%d", &matrix[i][j]);
if (matrix[i][j] != 0)
{
count++;
}
}
}
s_matrix *sparse_mat = (s_matrix *)malloc(count * sizeof(s_matrix));
int k = 0;
for(int i=0; i<m; i++)
{
for(int j=0; j<n; j++)
{
if(matrix[i][j] != 0)
{
sparse_mat[k].row = i;
sparse_mat[k].col = j;
sparse_mat[k].val = matrix[i][j];
k++;
}
}
}
printf("\nSparse Matrix Representation:\n");
printf("Row\tColumn\tValue\n");
for (int i = 0; i < count; i++)
{
printf("%d\t%d\t%d\n", sparse_mat[i].row, sparse_mat[i].col, sparse_mat[i].val);
}
for (int i = 0; i < m; i++)
{
free(matrix[i]);
}
free(matrix);
free(sparse_mat);
}
```
Problem 4: Dynamic Linked List Implementation
Objective: Implement a linked list using dynamic memory allocation.
Description:
1. Define a struct for linked list nodes. Each node should store an integer and a pointer to
the next node.
2. Create a menu-driven program to perform the following operations:
○ Add a node to the list.

○ Delete a node from the list.
○ Display the list.
3. Use malloc to allocate memory for each new node and free to deallocate memory for
deleted nodes.

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 rows, cols;
int **array;
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 values for the 2D array:\n");
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
printf("Enter value for array[%d][%d]: ", i, j);
scanf("%d", &array[i][j]);
}
```

```c
}
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);
printf("Memory freed successfully.\n");
return 0;
}
```

Problem 6: Student Record Management System

Objective

Create a program to manage student records using structures.

Requirements

1. Define a Student structure with the following fields:

char name[50]

int rollNumber

float marks

2. Implement functions to:

o Add a new student record.

Display all student records.

Find and display a student record by roll number.

Calculate and display the average marks of all students.

3. Implement a menu-driven interface to perform the above operations.

Output

```
1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit
Enter your choice: 1
Enter name: John Doe
Enter roll number: 101
Enter marks: 85.5
Student added successfully!
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_STUDENTS 100
```

```c
struct Student {
char name[50];
int rollNumber;
float marks;
};
// Global array to store student records
struct Student students[MAX_STUDENTS];
int studentCount = 0;
// Function to add a new student record
void addStudent() {
if (studentCount >= MAX_STUDENTS) {
printf("Student list is full. Cannot add more students.\n");
return;
}
struct Student newStudent;
printf("Enter name: ");
scanf(" %[^\n]s", newStudent.name); // Read name with spaces
printf("Enter roll number: ");
scanf("%d", &newStudent.rollNumber);
printf("Enter marks: ");
scanf("%f", &newStudent.marks);
students[studentCount] = newStudent;
studentCount++;
printf("Student added successfully!\n");
}
// Function to display all student records
void displayAllStudents() {
if (studentCount == 0) {
printf("No student records found.\n");
return;
}
printf("Student Records:\n");
for (int i = 0; i < studentCount; i++) {
printf("Name: %s, Roll Number: %d, Marks: %.2f\n",
students[i].name, students[i].rollNumber, students[i].marks);
}
}
// Function to find a student by roll number
void findStudentByRollNumber() {
int rollNumber;
printf("Enter roll number to search: ");
scanf("%d", &rollNumber);
for (int i = 0; i < studentCount; i++) {
if (students[i].rollNumber == rollNumber) {
```

```
printf("Student Found: Name: %s, Marks: %.2f\n",
students[i].name, students[i].marks);
return;
}
}
printf("Student with roll number %d not found.\n", rollNumber);
}
// Function to calculate and display the average marks of all students
void calculateAverageMarks() {
if (studentCount == 0) {
printf("No student records available to calculate average marks.\n");
return;
}
float totalMarks = 0;
for (int i = 0; i < studentCount; i++) {
totalMarks += students[i].marks;
}
float average = totalMarks / studentCount;
printf("Average Marks: %.2f\n", average);
}
int main() {
int choice;
while (1) {
printf("\nStudent Record Management System\n");
printf("1. Add Student\n");
printf("2. Display All Students\n");
printf("3. Find Student by Roll Number\n");
printf("4. Calculate Average Marks\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
addStudent();
break;
case 2:
displayAllStudents();
break;
case 3:
findStudentByRollNumber();
break;
case 4:
calculateAverageMarks();
break;
```

```
case 5:
printf("Exiting program.\n");
return 0;
default:
printf("Invalid choice. Please try again.\n");
}
}
}
```
Problem 1: Employee Management System
Objective: Create a program to manage employee details using structures.
Description:
1. Define a structure Employee with fields:
o int emp_id: Employee ID
o char name[50]: Employee name
o float salary: Employee salary
2. Write a menu-driven program to:
o Add an employee.
o Update employee salary by ID.
o Display all employee details.
o Find and display details of the employee with the highest salary.
```
#include <stdio.h>
struct Employee {
int emp_id;
char name[50];
float salary;
};
int main() {
struct Employee employees[100];
int count = 0, choice, id, i;
float max_salary;
int max_index;
do {
printf("\nMenu:\n");
printf("1. Add Employee\n");
printf("2. Update Salary by ID\n");
printf("3. Display All Employees\n");
printf("4. Find Employee with Highest Salary\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
if (choice == 1) {
printf("Enter Employee ID: ");
scanf("%d", &employees[count].emp_id);
printf("Enter Name: ");
scanf(" %[^\n]", employees[count].name);
printf("Enter Salary: ");
scanf("%f", &employees[count].salary);
count++;
}
else if (choice == 2) {
printf("Enter Employee ID to update salary: ");
scanf("%d", &id);
for (i = 0; i < count; i++) {
if (employees[i].emp_id == id) {
printf("Enter new salary: ");
scanf("%f", &employees[i].salary);
break;
}
}
}
else if (choice == 3) {
for (i = 0; i < count; i++) {
printf("ID: %d, Name: %s, Salary: %.2f\n",
employees[i].emp_id, employees[i].name, employees[i].salary);
}
}
else if (choice == 4) {
max_salary = employees[0].salary;
max_index = 0;
for (i = 1; i < count; i++) {
if (employees[i].salary > max_salary) {
max_salary = employees[i].salary;
max_index = i;
}
}
printf("Employee with Highest Salary: ID: %d, Name: %s, Salary: %.2f\n",
employees[max_index].emp_id, employees[max_index].name,
employees[max_index].salary);
}
} while (choice != 5);
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:
o int book_id: Book ID
o char title[100]: Book title

o char author[50]: Author name
o int copies: Number of available copies
2. Write a program to:
o Add books to the library.
o Issue a book by reducing the number of copies.
o Return a book by increasing the number of copies.
o 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;
};
int main() {
struct Book library[100];
int count = 0, choice;
do {
printf("\n1. Add Book\n2. Issue Book\n3. Return Book\n4. Search Book\n5.
Exit\nEnter
choice: ");
scanf("%d", &choice);
if (choice == 1) {
printf("Enter Book ID: ");
scanf("%d", &library[count].book_id);
printf("Enter Title: ");
scanf(" %[^\n]", library[count].title);
printf("Enter Author: ");
scanf(" %[^\n]", library[count].author);
printf("Enter Copies: ");
scanf("%d", &library[count].copies);
count++;
}
else if (choice == 2) {
int id;
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.\n");
break;
}
}
}
else if (choice == 3) {
int id;
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.\n");
break;
}
}
}
else if (choice == 4) {
char query[100];
printf("Enter Title or Author: ");
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);
}
}
}
} while (choice != 5);
return 0;
}
```

Problem 3: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. Define a structure Player with fields:

o char name[50]: Player name

o int matches: Number of matches played

o int runs: Total runs scored

o float average: Batting average

2. Write a program to:

o Input details for n players.

o Calculate and display the batting average for each player.

o 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() {
int n;
printf("Enter the number of players: ");
scanf("%d", &n);
struct Player players[n];
int highestIndex = 0;
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: ");
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 > players[highestIndex].average) {
highestIndex = i;
}
}
printf("\nPlayer Statistics:\n");
for (int i = 0; i < n; i++) {
printf("Name: %s, Matches: %d, Runs: %d, Average: %.2f\n",
players[i].name, players[i].matches, players[i].runs, players[i].average);
}
printf("\nPlayer with the highest batting average:\n");
printf("Name: %s, Average: %.2f\n", players[highestIndex].name,
players[highestIndex].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:
o int roll_no: Roll number
o char name[50]: Student name
o float marks[5]: Marks in 5 subjects
o char grade: Grade based on the average marks
2. Write a program to:

o Input details of n students.
o Calculate the average marks and assign grades (A, B, C, etc.).
o 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() {
int n;
printf("Enter the number of students: ");
scanf("%d", &n);
struct Student students[n];
for (int i = 0; i < n; i++) {
printf("Enter roll number and name for student %d: ", i + 1);
scanf("%d %[^\n]", &students[i].roll_no, students[i].name);
float total = 0;
printf("Enter marks for 5 subjects: ");
for (int j = 0; j < 5; j++) {
scanf("%f", &students[i].marks[j]);
total += students[i].marks[j];
}
float average = total / 5;
if (average >= 90) students[i].grade = 'A';
else if (average >= 75) students[i].grade = 'B';
else if (average >= 50) students[i].grade = 'C';
else students[i].grade = 'D';
}
printf("\nStudent Details:\n");
for (int i = 0; i < n; i++) {
printf("Roll No: %d, Name: %s, Grade: %c\n", students[i].roll_no, students[i].name,
students[i].grade);
}
return 0;
}
```

Problem 5: Flight Reservation System
Objective: Simulate a simple flight reservation system using structures.
Description:
1. Define a structure Flight with fields:
o char flight_number[10]: Flight number
o char destination[50]: Destination city
o int available_seats: Number of available seats

2. Write a program to:
o Add flights to the system.
o Book tickets for a flight, reducing available seats accordingly.
o Display the flight details based on destination.
o 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[5];
int flight_count = 0;
int choice;
do {
printf("\nFlight Reservation System Menu:\n");
printf("1. Add Flight\n");
printf("2. Book Ticket\n");
printf("3. Cancel Ticket\n");
printf("4. Display Flight Details\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
if (choice == 1) {
printf("Enter flight number: ");
scanf("%s", flights[flight_count].flight_number);
printf("Enter destination city: ");
scanf(" %[^\n]", flights[flight_count].destination);
printf("Enter number of available seats: ");
scanf("%d", &flights[flight_count].available_seats);
flight_count++;
}
else if (choice == 2) {
char flight_number[10];
printf("Enter flight number to book ticket: ");
scanf("%s", flight_number);
int found = 0;
for (int i = 0; i < flight_count; i++) {
if (strcmp(flights[i].flight_number, flight_number) == 0) {
if (flights[i].available_seats > 0) {
flights[i].available_seats--;
printf("Ticket booked successfully. Available seats: %d\n",
flights[i].available_seats);
} else {
printf("No available seats on this flight.\n");
}
found = 1;
break;
}
}
if (!found) {
printf("Flight not found.\n");
}
}
else if (choice == 3) {
char flight_number[10];
printf("Enter flight number to cancel ticket: ");
scanf("%s", flight_number);
int found = 0;
for (int i = 0; i < flight_count; i++) {
if (strcmp(flights[i].flight_number, flight_number) == 0) {
flights[i].available_seats++;
printf("Ticket canceled successfully. Available seats: %d\n",
flights[i].available_seats);
found = 1;
break;
}
}
if (!found) {
printf("Flight not found.\n");
}
}
else if (choice == 4) {
char destination[50];
printf("Enter destination city to display flight details: ");
scanf(" %[^\n]", destination);
int found = 0;
for (int i = 0; i < flight_count; i++) {
if (strcmp(flights[i].destination, destination) == 0) {
printf("Flight Number: %s\n", flights[i].flight_number);
printf("Destination: %s\n", flights[i].destination);
printf("Available Seats: %d\n", flights[i].available_seats);
found = 1;
}
}
if (!found) {
```

```
printf("No flights found to this destination.\n");
}
}
} while (choice != 5);
return 0;
}
```