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>


int main()

{

   int *array = NULL;

   int size, newsize;

   char choice;




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

   scanf("%d", &size);


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


   if (array == NULL) {

      printf("Memory allocation failed.\n");

      exit(0);
}
else
{
   printf("Memory is allocated\n");
}



printf("Enter  elements\n");
for (int i = 0; i < size; i++)
{
   scanf("%d", &array[i]);
}




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

printf("\n");



printf("Enter the new size : ");
scanf("%d", &newsize);

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

```
    if (newsize > size)
    {
        printf("Enter new elements:\n");
        for (int i = size; i < newsize; i++)
        {
            scanf("%d", &array[i]);
        }
    }
    printf("\n");

    printf("Array elements: ");
    for (int i = 0; i < newsize; i++) {
        printf("%d ", array[i]);
    }
}
```

Problem 2: String Concatenation Using Dynamic Memory

Objective: Create a program that concatenates two strings using dynamic memory allocation.

Description:

1. Accept two strings from the user.
2. Use malloc to allocate memory for the first string.
3. Use realloc to resize the memory to accommodate the concatenated string.
4. Concatenate the strings and print the result.
5. Free the allocated memory.

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

```
#include<string.h>

int main()
{
   char *str1 = NULL;
   char *str2 = NULL;
   int size1, size2;


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

   printf("Enter str1: ");
   scanf(" %s", str1);

   printf("\n");



   str2 = (char *)malloc(100 * sizeof(char));

   printf("Enter str2: ");
   scanf(" %s", str2);

   size1 = strlen(str1);
   size2 = strlen(str2);

   str1 = (char *)realloc(str1, (size1+size2) * sizeof(char));
```

```
    strcat(str1, str2);

    printf("\n");

    printf("Concatenated string: %s\n", str1);



    free(str1);
    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.

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

struct sparse_matrix {
    int row;
    int col;
    int val;
};

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++;
            }
        }
    }
```

```
  struct sparse_matrix *sparse_mat = (struct sparse_matrix *)malloc(count *
sizeof(struct sparse_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);

    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.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 int row,col;
 printf("Enter the values for row and column: ");
 scanf("%d %d",&row,&col);
 int matrix[row][col];
 int *ptr1 = NULL;
 int *ptr2 = NULL;
 ptr1 = (int *)malloc(row*sizeof(int));
 ptr2 = (int *)malloc(col*sizeof(int));
 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");
 }
 free(ptr1);
 free(ptr2);

 return 0;

}
```

6- Student management

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

```
struct students

{

   char name[50];

   int rollno;

   float marks;

};


struct students array[100];

int count=0;


void addstudent();

void printstudent();

void findstudent();

void averagemark();


int main()

{

   int a=0;

   int choice;

   while(a!=1)

   {


      printf(" 1.Add student\n 2.Display all Students\n 3.Find student ny rollno\n
4.Calculate avg mark\n 5.Exit\n");

      printf("Enter the choice: ");

      scanf("%d",&choice);

      switch(choice)
      {
        case 1:
          addstudent();
          break;

        case 2:
          printstudent();
          break;

        case 3:
          findstudent();
          break;

        case 4:
          averagemark();
          break;

        case 5:
          a=1;
          break;
      }
    }
    return 0;
}


void addstudent()
{
    printf("\n");

    printf("Enter student name: ");

    scanf("%s",array[count].name);

    printf("Enter student rollno: ");

    scanf("%d",&array[count].rollno);

    printf("Enter student mark: ");

    scanf("%f",&array[count].marks);


    printf("Details Addeed\n");

    printf("\n");


    count =count+1;

}



void printstudent()

{

    printf("\n");

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

    {

        printf("Name: %s Rollno: %d Mark: %.2f\n",array[i].name, array[i].rollno,
array[i].marks);

    }

    printf("\n");

}



void findstudent()
```

```
{
   int rollno;
   int found=0;
   char name1[10];
   printf("\n");
   printf("Enter roll number to search: ");
   scanf("%d", &rollno);
   for (int i = 0; i < count; i++)
   {
      if (array[i].rollno == rollno)
      {
         found=1;
         strcpy(name1,array[i].name);

      }
   }
   if(found==1)
   {
      printf("Student Found:\n");
      printf("Name is %s",name1);
      printf("\n");
   }
   else
   {
      printf("Student Not Found");
      printf("\n");
   }
}
```

```
void averagemark()
{
  float total=0;
  float total_marks=0;
  for (int i = 0; i < count; i++)
  {
    total_marks += array[i].marks;
  }
  float average = total_marks / count;
  printf("\n");
  printf("The average marks of students : %f\n", average);
}
```

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>

struct employee
{
   int empid;
   char name[50];
   float salary;
};
struct employee array[100];
int count=0;

void addemployee();
void updateemployee();
void displayemployee();
void findsalary();

int main()
{
   int a=0;
   int choice;
   while(a!=1)
   {

      printf("1.Add employee\n2.Update emp sal by id\n3.Display all employe\n4.Display
details of emp with highest sal\n5.Exit\n");
      printf("Enter the choice: ");
      scanf("%d",&choice);
```

```
        switch(choice)

        {

          case 1:

            addemployee();

            break;


          case 2:

            updateemployee();

            break;


          case 3:

            displayemployee();

            break;


          case 4:

            findsalary();

            break;


          case 5:

            a=1;

            break;

        }

    }

    return 0;

}



void addemployee()
```

```
{
   printf("\n");
   printf("Enter employee name: ");
   scanf("%s",array[count].name);

   printf("Enter employee id: ");
   scanf("%d",&array[count].empid);

   printf("Enter salary: ");
   scanf("%f",&array[count].salary);

   printf("Details Addeed\n");
   printf("\n");

   count =count+1;
}

void updateemployee()
{
   int id;
   int found=0;
   printf("Enter employee id: ");
   scanf("%d",&id);
   for(int i=0;i<count;i++)
   {
      if(id==array[i].empid)
      {
         printf("Enter the new salary: ");
```

```
            scanf("%f",&array[i].salary);

            printf("\n");

            found=1;

        }


    }

    if(found==0)

    {

        printf("No employee found");

        printf("\n");

    }

}


void displayemploye()

{

    printf("\n");

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

    {

        printf("Name: %s Empid: %d Salary: %.2f\n",array[i].name, array[i].empid,
array[i].salary);

    }

    printf("\n");

}


void findsalary()

{

    float max;

    int flag;
```

```
  max=array[0].salary;
  for(int i=1;i<count;i++)
  {
    if(array[i].salary>max)
    {
      max=array[i].salary;
      flag=i;
    }
  }
  printf("Name: %s Empid: %d Salary: %.2f\n",array[flag].name, array[flag].empid,
array[flag].salary);
  printf("\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>

struct book
{
   int book_id;
   char title[100];
   char author[50];
   int copies;
};
struct book library[100];
int count = 0;

void addbook();
void issuebook();
void returnbook();
void searchbook();

int main()
{
   int a = 0;
   int choice;
   while(a != 1)
   {
      printf("1. Add book\n2. Issue book\n3. Return book\n4. Search book\n5. Exit\n");
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch(choice)
      {
        case 1:
          addbook();
          break;

        case 2:
          issuebook();
          break;

        case 3:
          returnbook();
          break;

        case 4:
          searchbook();
          break;

        case 5:
          a = 1;
          break;
      }
    }
    return 0;
}

void addbook()
{
    printf("\nEnter book title: ");
    scanf(" %[^\n]", library[count].title);
    printf("Enter book author: ");
    scanf(" %[^\n]", library[count].author);

    printf("Enter book id: ");
    scanf("%d", &library[count].book_id);

    printf("Enter number of copies: ");
    scanf("%d", &library[count].copies);

    printf("Book Added\n\n");
    count = count + 1;
}

void issuebook()
{
    int id;
    int found = 0;
    printf("Enter book id to issue: ");
    scanf("%d", &id);
    for(int i = 0; i < count; i++)
    {
        if(id == library[i].book_id)
        {
            if(library[i].copies > 0)
            {
                library[i].copies--;
                printf("Book Issued. Remaining copies: %d\n", library[i].copies);
        }
        else
        {
          printf("No copies available to issue\n");
        }
        found = 1;
        break;
      }
    }
    if(found == 0)
    {
      printf("No book found with that ID\n");
    }
}

void returnbook()
{
    int id;
    int found = 0;
    printf("Enter book id to return: ");
    scanf("%d", &id);
    for(int i = 0; i < count; i++)
    {
      if(id == library[i].book_id)
      {
        library[i].copies++;
        printf("Book Returned. Total copies: %d\n", library[i].copies);
        found = 1;
```

```
            break;
        }
    }
    if(found == 0)
    {
        printf("No book found with that ID\n");
    }
}

void searchbook()
{
    int choice;
    printf("Search book by:\n1. Title\n2. Author\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    getchar();
    if(choice == 1)
    {
        char title[100];
        int found = 0;
        printf("Enter book title: ");
        scanf(" %[^\n]", title);
        for(int i = 0; i < count; i++)
        {
            if(strcmp(library[i].title, title) == 0)
            {
                printf("Book found: ID: %d Author: %s Copies: %d\n", library[i].book_id,
library[i].author, library[i].copies);
```

```
            found = 1;

          }

        }

        if(found == 0)

        {

          printf("No book found with that title\n");

        }

      }

      else if(choice == 2)

      {

        char author[50];

        int found = 0;

        printf("Enter author name: ");

        scanf(" %[^\n]", author);

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

        {

          if(strcmp(library[i].author, author) == 0)

          {

            printf("Book found: ID: %d Title: %s Copies: %d\n", library[i].book_id,
library[i].title, library[i].copies);

            found = 1;

          }

        }

        if(found == 0)

        {

          printf("No book found by that author\n");

        }

      }
```

```
  else
  {
     printf("Invalid choice\n");
  }
}
```

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;
};
```

```
struct player array[100];

int count = 0;


void addplayer();

void calculateaverage();

void highestaverage();


int main()

{

  int a = 0;

  int choice;

  while(a != 1)

  {

    printf("1. Add player\n2. Calculate batting average\n3. Find player with highest
batting average\n4. Exit\n");

    printf("Enter the choice: ");

    scanf("%d", &choice);

    switch(choice)

    {

      case 1:

        addplayer();

        break;


      case 2:

        calculateaverage();

        break;


      case 3:
            highestaverage();
            break;

         case 4:
            a = 1;
            break;
      }
   }
   return 0;
}

void addplayer()
{
   printf("\nEnter player name: ");
   scanf(" %[^\n]", array[count].name);

   printf("Enter number of matches played: ");
   scanf("%d", &array[count].matches);

   printf("Enter total runs scored: ");
   scanf("%d", &array[count].runs);

   count = count + 1;
   printf("Player Added\n\n");
}

void calculateaverage()
{
  for(int i = 0; i < count; i++)
  {
    if(array[i].matches > 0)
    {
      array[i].average = (float)array[i].runs / array[i].matches;
      printf("Player: %s, Batting Average: %.2f\n", array[i].name, array[i].average);
    }
    else
    {
      printf("Player: %s has played 0 matches. Cannot calculate average.\n",
array[i].name);
    }
  }
  printf("\n");
}


void highestaverage()
{
  if(count == 0)
  {
    printf("No players available.\n");
    return;
  }

  int flag = 0;
  float maxAverage = array[0].average;
  int maxIndex = 0;
  for(int i = 1; i < count; i++)
  {
    if(array[i].average > maxAverage)
    {
      maxAverage = array[i].average;
      maxIndex = i;
    }
  }

  printf("Player with highest batting average: %s, Average: %.2f\n",
array[maxIndex].name, array[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;
   float average;
};
struct student students[100];
int count = 0;

void addstudent();
void calculateresults();
void displaystudents();

int main()
{
   int a = 0;
   int choice;
   while(a != 1)
   {
      printf("1. Add student\n2. Calculate average and assign grades\n3. Display all
students\n4. Exit\n");
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch(choice)
      {
         case 1:
           addstudent();
           break;

         case 2:
           calculateresults();
           break;

         case 3:
           displaystudents();
           break;

         case 4:
           a = 1;
           break;

      }
   }
   return 0;
}

void addstudent()
{
   printf("\nEnter student roll number: ");
   scanf("%d", &students[count].roll_no);

   printf("Enter student name: ");
   scanf(" %[^\n]", students[count].name);
   printf("Enter marks in 5 subjects: ");
   for(int i = 0; i < 5; i++)
   {
      scanf("%f", &students[count].marks[i]);
   }

   count++;
   printf("Student Added\n\n");
}

void calculateresults()
{
   for(int i = 0; i < count; i++)
   {
      float total = 0;
      for(int j = 0; j < 5; j++)
      {
           total += students[i].marks[j];
      }
      students[i].average = total / 5;



      if(students[i].average >= 90)
           students[i].grade = 'A';
      else if(students[i].average >= 75)
           students[i].grade = 'B';
      else if(students[i].average >= 50)
           students[i].grade = 'C';
      else
      students[i].grade = 'F';
   }
   printf("Grades calculated for all students.\n\n");
}

void displaystudents()
{
   printf("\nList of students:\n");
   for(int i = 0; i < count; i++)
   {
      printf("Roll No: %d, Name: %s, Average Marks: %.2f, Grade: %c\n",
           students[i].roll_no, students[i].name, students[i].average, students[i].grade);
   }
   printf("\n");
}
```

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;
};
struct flight flights[100];
int count = 0;

void addflight();
void bookticket();
void displayflights();
void cancelticket();

int main()
{
   int a = 0;
   int choice;
   while(a != 1)
   {
      printf("1. Add flight\n2. Book ticket\n3. Display flights by destination\n4. Cancel
ticket\n5. Exit\n");
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch(choice)
      {
        case 1:
          addflight();
          break;

        case 2:
          bookticket();
          break;

        case 3:
          displayflights();
          break;

        case 4:
          cancelticket();
          break;

        case 5:
          a = 1;
          break;



      }
    }
    return 0;
  }

  void addflight()
```

```
{
    printf("\nEnter flight number: ");
    scanf("%s", flights[count].flight_number);

    printf("Enter destination: ");
    scanf(" %[^\n]", flights[count].destination);

    printf("Enter number of available seats: ");
    scanf("%d", &flights[count].available_seats);

    count++;
    printf("Flight Added\n\n");
}

void bookticket()
{
    char flight_no[10];
    int found = 0;

    printf("\nEnter flight number: ");
    scanf("%s", flight_no);

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

```
            flights[i].available_seats--;
            printf("Ticket booked successfully.");
          }
          else
          {
            printf("No seats available.\n");
          }
          found = 1;
          break;
        }
    }
    if(!found)
    {
        printf("Flight not found.\n\n");
    }
}

void displayflights()
{
    char dest[50];
    int found = 0;

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

void cancelticket()
{
  char flight_no[10];
  int found = 0;

  printf("\nEnter flight number: ");
  scanf("%s", flight_no);

  for(int i = 0; i < count; i++)
  {
    if(strcmp(flight_no, flights[i].flight_number) == 0)
    {
      flights[i].available_seats++;
      printf("Ticket canceled successfully.\n");
      found = 1;
```

```
            break;
        }
    }
    if(!found)
    {
        printf("Flight not found.\n");
    }
}
```