3.

```c
/*
Problem 1: Book Inventory System
Problem Statement:
Write a C program to manage a book inventory system using dynamic memory
allocation. The program should:
1.  Define a structure named Book with the following fields:
o   id (integer): The book's unique identifier.
o   title (character array of size 100): The book's title.
o   price (float): The price of the book.
2.  Dynamically allocate memory for n books (where n is input by the user).
3.  Implement the following features:
o   Input Details: Input details for each book (ID, title, and price).
o   Display Details: Display the details of all books.
o   Find Cheapest Book: Identify and display the details of the cheapest book.
o   Update Price: Allow the user to update the price of a specific book by
entering its ID.


*/

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

struct Book{
    int id;
    char title[100];
    float price;

};


void insertDetails(struct Book *,int);
void displayDetails(struct Book *,int);
void cheapestBook(struct Book *,int);
void updatePrice(struct Book *,int);

int main(){
    int n,choice;
    struct Book *ptr;


    printf("\nEnter the number of books :");
    scanf("%d",&n);

    ptr = (struct Book *)malloc(n*(sizeof(struct Book)));
    if (ptr == NULL) {
        printf("Memory allocation failed. Exiting.\n");
        return 1;
    }

     while (1) {
        printf("\nChoose from the options:\n");
        printf("1. Insert Details\n2. Display Details\n3. Find Cheapest
Book\n4. Update Price\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertDetails(ptr, n);
                break;
            case 2:
                displayDetails(ptr, n);
                break;
            case 3:
                cheapestBook(ptr, n);
                break;
            case 4:
                updatePrice(ptr, n);
                break;
            case 5:
                free(ptr);  // Free allocated memory before exiting
                printf("Exiting program. Goodbye!\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}

void insertDetails(struct Book *ptr,int n){
    //1.Input the details from user

    printf("\nEnter the details of books:\n");
    for (int i = 0; i < n; i++) {
        printf("Enter ID, Title, and Price of Book %d:\n", i + 1);
        printf("ID: ");
        scanf("%d", &(ptr + i)->id);

        // Check for duplicate IDs
        for (int j = 0; j < i; j++) {
            if ((ptr + j)->id == (ptr + i)->id) {
                printf("Invalid ID: ID already exists! Enter a unique ID.\n");
                i--;  // Decrement `i` to allow re-entering this book
                break;
            }
        }
        printf("\nTitle : ");
        scanf("%s",(ptr+i)->title);
        printf("\nPrice : ");
        scanf("%f",&(ptr+i)->price);
}
}
void displayDetails(struct Book *ptr,int n)
{
    printf("\nDetails of Books!!!");
    for(int i=0;i<n;i++)
    {
        printf("\n%d\t%s\t%.1f",(ptr+i)->id,(ptr+i)->title,(ptr+i)->price);
    }
}
void cheapestBook(struct Book *ptr,int n){
    int temp = 0;
    for(int i=0;i<n;i++)
    {

        if(((ptr+i)->price)<((ptr+temp)->price))
        {
            temp = i;
        }

    }
    printf("\nThe Details of the cheapest book is given below.");
    printf("\n%d\t%s\t%.1f",(ptr+temp)->id,(ptr+temp)->title,(ptr+temp)-
>price);
}
void updatePrice(struct Book *ptr,int n){
    int num;
    bool found = false;
    printf("\nEnter id of book whose is price you want to update :");
    scanf("%d",&num);

    for(int i=0;i<n;i++)
    {
        if((ptr+i)->id==num){
            found = true;
            printf("\nEnter new price : ");
            scanf("%f",&(ptr+i)->price);
        }
    }
    if(!found)
    {
        printf("\nId not found!!");
    }

}
```

OUTPUT :

```
Enter the number of books :2


Choose from the options:

1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 1


Enter the details of books:

Enter ID, Title, and Price of Book 1:

ID: 1


Title : ABCD


Price : 250

Enter ID, Title, and Price of Book 2:

ID: 2


Title : Songs
```

```
Price : 150


Choose from the options:

1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 2


Details of Books!!!

1       ABCD    250.0

2       Songs   150.0

Choose from the options:

1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 3


The Details of the cheapest book is given below.

2       Songs   150.0

Choose from the options:
```

```
1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 4


Enter id of book whose is price you want to update :4


Id not found!!

Choose from the options:

1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 4


Enter id of book whose is price you want to update :2


Enter new price : 200


Choose from the options:

1. Insert Details

2. Display Details
```

```
3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 2


Details of Books!!!

1       ABCD    250.0

2       Songs   200.0

Choose from the options:

1. Insert Details

2. Display Details

3. Find Cheapest Book

4. Update Price

5. Exit

Enter your choice: 5

Exiting program. Goodbye!
```

1.

```c
//Accessing structure member through pointer using dynamic memory allocation
#include <stdio.h>
#include <stdlib.h>

struct course{
    int marks;
    char subject[30];
};




int main(){
    struct course *ptr;
    int noOfRecords;
    printf("\nEnter the number of records : ");
    scanf("%d",&noOfRecords);


    //Dynamic memory allocation for number of allocations

    ptr = (struct course *)malloc(noOfRecords*(sizeof(struct course)));//Where
(struct course *) is the type casting part in this dynamic memory allocation


    for(int i=0;i<noOfRecords;i++)
    {
        printf("\nEnter Subject name and mark : ");
        scanf("%s %d",(ptr+i)->subject,&(ptr+i)->marks);
    }


    //Display Informations

    printf("\nDisplaying entered Informations : \n");
    for(int i=0;i<noOfRecords;i++){
        printf("\n%s\t%d",(ptr+i)->subject,(ptr+i)->marks);
    }
    free(ptr);

    return 0;
}
```

2.

```c
/*Problem Statement: Employee Records Management
Write a C program to manage a list of employees using dynamic memory
allocation. The program should:
Define a structure named Employee with the following fields:
id (integer): A unique identifier for the employee.
name (character array of size 50): The employee's name.
salary (float): The employee's salary.
Dynamically allocate memory for storing information about n employees (where n
is input by the user).
Implement the following features:
Input Details: Allow the user to input the details of each employee (ID, name,
and salary).
Display Details: Display the details of all employees.
Search by ID: Allow the user to search for an employee by their ID and display
their details.
```

```
Free Memory: Ensure that all dynamically allocated memory is freed at the end
of the program.

Constraints
n (number of employees) must be a positive integer.
Employee IDs are unique.

Sample Input/Output
Input:
Enter the number of employees: 3

Enter details of employee 1:
ID: 101
Name: Alice
Salary: 50000

Enter details of employee 2:
ID: 102
Name: Bob
Salary: 60000

Enter details of employee 3:
ID: 103
Name: Charlie
Salary: 55000

Enter ID to search for: 102
Output:
Employee Details:
ID: 101, Name: Alice, Salary: 50000.00
ID: 102, Name: Bob, Salary: 60000.00
ID: 103, Name: Charlie, Salary: 55000.00

Search Result:
ID: 102, Name: Bob, Salary: 60000.00*/

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

struct Employee
{
    int id;
    char name[50];
    float salary;
};


int main(){
```

```c
    int n;
    struct Employee *ptr;

    printf("\nEnter the number of employee's :");
    scanf("%d",&n);

    //Dynamically allocating memory for n employee's

    ptr = (struct Employee *)malloc(n*(sizeof(struct Employee)));

    //Features Implementation
    //1.Input the details from user

    printf("\nEnter the Details of Employees's \n");
    for(int i=0;i<n;i++){
        printf("\nEnter the id,name,salary of Employee %d :",i+1);
        scanf("%d %s %f",&(ptr+i)->id,(ptr+i)->name,&(ptr+i)->salary);

    }/////take care of id cannot be repeated?????????

    //2.Display the details of the employees
    printf("\nThe details of employees : \n");
    for(int i=0;i<n;i++)
    {
        printf("\n%d\t%s\t%.2f",(ptr+i)->id,(ptr+i)->name,(ptr+i)->salary);
    }


    //3.Search by ID
    int id;
    printf("\nEnter the ID of Employee : ");
    scanf("%d",&id);
    bool found = false;

    for(int i=0;i<n;i++)
    {
        if(id==(ptr+i)->id)
        {
            found = true;
            printf("The details of the employee is
here!\n%d\t%s\t%.2f",(ptr+i)->id,(ptr+i)->name,(ptr+i)->salary);
        }

    }
    if(!found)
    {
        printf("\nEmployee Details not Found!!!");

    }
    free(ptr);

    return 0;
}
```

4.

```c
//Union DS Concepts
//Accessing structure member through pointer using dynamic memory allocation
#include <stdio.h>
#include <stdlib.h>

struct course{
    int marks;
    char subject[30];
};

union course1
{
    int marks;
    char subject[30];
};



int main(){

    struct course strVar;
    union course1 uniVar;

    printf("strVar = %d, uniVar = %d",sizeof(strVar),sizeof(uniVar));


    return 0;
}
```

5.

```c
/*Problem 2: Dynamic Point Array
Problem Statement:
Write a C program to handle a dynamic array of points in a 2D space using
dynamic memory allocation. The program should:
1.  Define a structure named Point with the following fields:
o   x (float): The x-coordinate of the point.
o   y (float): The y-coordinate of the point.
```

```c
2.  Dynamically allocate memory for n points (where n is input by the user).
3.  Implement the following features:
o   Input Details: Input the coordinates of each point.
o   Display Points: Display the coordinates of all points.
o   Find Distance: Calculate the Euclidean distance between two points chosen
by the user (by their indices in the array).
o   Find Closest Pair: Identify and display the pair of points that are
closest to each other.
*/


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

struct Point{
    float x;
    float y;
};


void insertDetails(struct Book *,int);
void displayDetails(struct Book *,int);
void findDistance(struct Book *,int);
void cPair(struct Book *,int);

int main()
{
    int n;
    printf("\nEnter the number of points :");
    scanf("%d",&n);

    struct Point *ptr = (struct Point *)malloc(n*(sizeof(struct Point)));


    if (ptr == NULL) {
        printf("Memory allocation failed. Exiting.\n");
        return 1;
    }

     while (1) {
        printf("\nChoose from the options:\n");
        printf("1. Input Details\n2. Display Points\n3. Find Distance\n4. Find
Closest Pair\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertDetails(ptr, n);
                break;
            case 2:
                displayDetails(ptr, n);
                break;
            case 3:
                findDistance(ptr, n);
                break;
            case 4:
                cPair(ptr, n);
                break;
            case 5:
                free(ptr);  // Free allocated memory before exiting
                printf("Exiting program. Goodbye!\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
void insertDetails(struct Book *ptr,int n){
    //1.Input the details from user

    printf("\nEnter the points of the coordinate:\n");
    for (int i = 0; i < n; i++) {
        printf("Enter X - Coordinate of  %d point: ", i + 1);

        scanf("%f", &(ptr + i)->x);

        // Check for duplicate IDs

        printf("Enter Y - Coordinate of  %d point: ", i + 1);

        scanf("%f", &(ptr + i)->y);
}
}
void displayDetails(struct Book *ptr,int n)
{
    printf("\nDetails of Books!!!");
    for(int i=0;i<n;i++)
    {
        printf("\nX : %.1f\tY : %.1f",(ptr+i)->x,(ptr+i)->y);
    }
}
void findDistance(struct Point *ptr, int n) {
    int index1, index2;
    printf("\nEnter the indices of the two points (1 to %d) to calculate the
distance:\n", n);
```

```c
    printf("First point index: ");
    scanf("%d", &index1);
    printf("Second point index: ");
    scanf("%d", &index2);

    // Adjust indices to zero-based
    index1--;
    index2--;

    if (index1 < 0 || index2 < 0 || index1 >= n || index2 >= n) {
        printf("Invalid indices. Please try again.\n");
        return;
    }

    float distance = sqrt(pow(ptr[index2].x - ptr[index1].x, 2) +
                          pow(ptr[index2].y - ptr[index1].y, 2));
    printf("The distance between Point %d and Point %d is: %.2f\n", index1 +
1, index2 + 1, distance);
}

// Function to find the closest pair of points
void cPair(struct Point *ptr, int n) {
    if (n < 2) {
        printf("Not enough points to find a closest pair.\n");
        return;
    }

    int p1 = 0, p2 = 1;
    float minDistance = sqrt(pow(ptr[1].x - ptr[0].x, 2) +
                             pow(ptr[1].y - ptr[0].y, 2));

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            float distance = sqrt(pow(ptr[j].x - ptr[i].x, 2) +
                                  pow(ptr[j].y - ptr[i].y, 2));
            if (distance < minDistance) {
                minDistance = distance;
                p1 = i;
                p2 = j;
            }
        }
    }

    printf("The closest pair of points is:\n");
    printf("Point %d: (%.2f, %.2f)\n", p1 + 1, ptr[p1].x, ptr[p1].y);
    printf("Point %d: (%.2f, %.2f)\n", p2 + 1, ptr[p2].x, ptr[p2].y);
    printf("Their distance is: %.2f\n", minDistance);
}
```

6.

```
/*
Problem Statement: Vehicle Registration System
Write a C program to simulate a vehicle registration system using unions to
handle different types of vehicles. The program should:
1.  Define a union named Vehicle with the following members:
o   car_model (character array of size 50): To store the model name of a car.
o   bike_cc (integer): To store the engine capacity (in CC) of a bike.
o   bus_seats (integer): To store the number of seats in a bus.
2.  Create a structure VehicleInfo that contains:
o   type (character): To indicate the type of vehicle (C for car, B for bike,
S for bus).
o   Vehicle (the union defined above): To store the specific details of the
vehicle based on its type.
3.  Implement the following features:
o   Input Details: Prompt the user to input the type of vehicle and its
corresponding details:
♣   For a car: Input the model name.
♣   For a bike: Input the engine capacity.
♣   For a bus: Input the number of seats.
o   Display Details: Display the details of the vehicle based on its type.
4.  Use the union effectively to save memory and ensure only relevant
information is stored.

Constraints
•   The type of vehicle should be one of C, B, or S.
•   For invalid input, prompt the user again.









Sample Input/Output

Input:
```

```
Enter vehicle type (C for Car, B for Bike, S for Bus): C
Enter car model: Toyota Corolla
Output:
Vehicle Type: Car
Car Model: Toyota Corolla


Input:
Enter vehicle type (C for Car, B for Bike, S for Bus): B
Enter bike engine capacity (CC): 150
Output:
Vehicle Type: Bike
Engine Capacity: 150 CC

Input:
Enter vehicle type (C for Car, B for Bike, S for Bus): S
Enter number of seats in the bus: 50
Output:
Vehicle Type: Bus
Number of Seats: 50





*/
#include <stdio.h>
#include <string.h>

// Define a union to store vehicle-specific details
union Vehicle {
    char car_model[50]; // For car model name
    int bike_cc;        // For bike engine capacity
    int bus_seats;      // For bus seating capacity
};

// Define a structure to hold vehicle information
struct VehicleInfo {
    char type;          // Type of vehicle: C, B, or S
    union Vehicle details; // Union for vehicle-specific details
};

// Function to input details of the vehicle
void inputDetails(struct VehicleInfo *vehicle) {
    printf("Enter vehicle type (C for Car, B for Bike, S for Bus): ");
    while (1) {
        scanf(" %c", &vehicle->type);
        if (vehicle->type == 'C' || vehicle->type == 'B' || vehicle->type ==
'S') {
```

```
            break;
        } else {
            printf("Invalid input. Please enter C, B, or S: ");
        }
    }

    // Input details based on vehicle type
    switch (vehicle->type) {
        case 'C':
            printf("Enter car model: ");
            scanf(" %[^\n]", vehicle->details.car_model); // Input car model
            break;
        case 'B':
            printf("Enter bike engine capacity (CC): ");
            scanf("%d", &vehicle->details.bike_cc); // Input bike engine
capacity
            break;
        case 'S':
            printf("Enter number of seats in the bus: ");
            scanf("%d", &vehicle->details.bus_seats); // Input number of seats
            break;
    }
}

// Function to display the vehicle details
void displayDetails(const struct VehicleInfo *vehicle) {
    printf("Vehicle Type: ");
    switch (vehicle->type) {
        case 'C':
            printf("Car\n");
            printf("Car Model: %s\n", vehicle->details.car_model);
            break;
        case 'B':
            printf("Bike\n");
            printf("Engine Capacity: %d CC\n", vehicle->details.bike_cc);
            break;
        case 'S':
            printf("Bus\n");
            printf("Number of Seats: %d\n", vehicle->details.bus_seats);
            break;
    }
}

// Main function
int main() {
    struct VehicleInfo vehicle;

    inputDetails(&vehicle);    // Input the vehicle details
    displayDetails(&vehicle);  // Display the vehicle details

    return 0;
}
```

7.

```
//Union DS Concepts

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


union course1
{
    int a;
    int b;
}Var;



int main(){



    Var.a = 10;


    printf("1...a = %d, b = %d",Var.a,Var.b);


    Var.b = 20;
    printf("\n2...a = %d, b = %d",Var.a,Var.b);//We are getting same output
for both a and b because both uses the same address
    return 0;
}
```

8.

```
//Union DS Concepts
//Pointer in Union
#include <stdio.h>
#include <stdlib.h>
```

```c
union test
{
    int a;
    int b;
}Var;



int main(){


    union test *ptr;
    ptr = &Var;

    ptr->a = 10;


    printf("\n1....a = %d, b = %d",ptr->a,ptr->b);


    ptr->b = 20;
    printf("\n2...a = %d, b = %d",ptr->a,ptr->b);//We are getting same output
for both a and b because both uses the same address
    return 0;
}
```

9.

```c
//Union DS Concepts
//fn in Union
#include <stdio.h>
#include <stdlib.h>


union test
{
    int a;
    int b;
};

void add(union test );

int main(){


    union test var;


    var.a = 10;

    printf("\n1....a = %d, b = %d",var.a,var.b);

    add(var);

    var.b = 20;
    printf("\n2...a = %d, b = %d",var.a,var.b);//We are getting same output for both a and b because both uses the same address
    add(var);
    return 0;
}

void add(union test var)
{
    int sum =0;
    sum = var.a+var.b;
    printf("\nSum = %d",sum);
}
```

10.

```c
//Union DS Concepts
//fn in Union
//Passing Pointer to fn


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


union test
{
    int a;
    int b;
}var;

void add(union test *);

int main(){


    union test *ptr = &var;



    var.a = 10;
    printf("\n1....a = %d, b = %d",var.a,var.b);

    add(ptr);

    var.b = 20;
    printf("\n2...a = %d, b = %d",var.a,var.b);//We are getting same output
for both a and b because both uses the same address
    add(ptr);
    return 0;
}

void add(union test *ptr1)
{
    int sum =0;
    sum = ptr1->a+ptr1->b;
    printf("\nSum = %d",sum);
}
```

11.

```c
//Enumerator concept
//Size of enum is 4 byte
#include <stdio.h>

enum math{
    add = 1,
    sub
};


int main(){
  enum math var1 = add;
  printf("size of var1 = %d \n",sizeof(var1));
  switch(var1){
      case 1:
      printf("Addiiotn opration\n");
      break;
      case 2:
      printf("Substraction Operation\n");
      break;
      case 3:
      printf("Division Opeartion\n");
      break;
      default:
      printf("Wrong Option\n");
      break;
  }
    return 0;
}
```

12.

```
/*
Problem 1: Traffic Light System
Problem Statement:
Write a C program to simulate a traffic light system using enum. The program
should:
1.   Define an enum named TrafficLight with the values RED, YELLOW, and GREEN.
2.   Accept the current light color as input from the user (as an integer: 0
for RED, 1 for YELLOW, 2 for GREEN).
3.   Display an appropriate message based on the current light:
o    RED: "Stop"
o    YELLOW: "Ready to move"
o    GREEN: "Go"



*/
#include <stdio.h>

enum math{
    RED,
    YELLOW,
    GREEN
};


int main(){

  int i;
  printf("Current Light : ");
  scanf("%d",&i);
  if(i<0||i>2)
  {
    printf("\nInvalid Input!!!Please choose from 0,1 or 2");
    return 1;
  }
  enum math var1 = i;
  switch(var1){
      case 0:
      printf("RED: Stop\n");
      break;
      case 1:
```

```
      printf("YELLOW: Ready to move\n");
      break;
      case 2:
      printf("GREEN: Go\n");
      break;
      default:

      break;
  }
    return 0;
}
```

13.

```
/*Problem 2: Days of the Week
Problem Statement:
Write a C program that uses an enum to represent the days of the week. The
program should:
1.  Define an enum named Weekday with values MONDAY, TUESDAY, WEDNESDAY,
THURSDAY, FRIDAY, SATURDAY, and SUNDAY.
2.  Accept a number (1 to 7) from the user representing the day of the week.
3.  Print the name of the day and whether it is a weekday or a weekend.
o   Weekends: SATURDAY and SUNDAY
o   Weekdays: The rest
*/
#include <stdio.h>

enum Weekday
{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
};


int main()
{
    int i;
    printf("\nEnter input number that represent the day of a week :");
    scanf("%d",&i);

    if(i<1 || i>7)
```

```c
    {
        printf("\nINVALID INPUT!! Please choose from (1 to 7).");
        return 1;
    }


    enum Weekday var = i;


    switch (var)
    {
    case 1:
        printf("MONDAY\nWEEKDAY");
        break;
    case 2:
        printf("TUESDAY\nWEEKDAY");
        break;
    case 3:
        printf("WEDNESDAY\nWEEKDAY");
        break;
    case 4:
        printf("THURSDAY\nWEEKDAY");
        break;
    case 5:
        printf("FRIDAY\nWEEKDAY");
        break;
    case 6:
        printf("SATURDAY\nWEEKEND");
        break;
    case 7:
        printf("SUNDAY\nWEEKEND");
        break;
    default:

        break;
    }

}
```

14.

```c
/*
Problem 3: Shapes and Their Areas
Problem Statement:
Write a C program to calculate the area of a shape based on user input using
enum. The program should:
1.  Define an enum named Shape with values CIRCLE, RECTANGLE, and TRIANGLE.
```

```c
2.  Prompt the user to select a shape (0 for CIRCLE, 1 for RECTANGLE, 2 for
TRIANGLE).
3.  Based on the selection, input the required dimensions:
o   For CIRCLE: Radius
o   For RECTANGLE: Length and breadth
o   For TRIANGLE: Base and height
4.  Calculate and display the area of the selected shape.
*/


#include <stdio.h>

enum Shape{
    CIRCLE,
    RECTANGLE,
    TRIANGLE
};

int main(){
    int i;
    printf("Select the Shape : ");
    scanf("%d",&i);
    if(i<0||i>2)
    {
        printf("\nInvalid Input!!!Please choose from 0,1 or 2");
        return 1;
    }
    enum Shape var = i;
    switch (var)
    {
    case 0 : printf("Enter the CIRCLE Radius : ");
             int r;
             scanf("%d",&r);
             float area = 3.14 * r * r;
             printf("\nArea of Circle is %.1f",area);
             break;
    case 1 : printf("Enter the RECTANGLE Length and breadth : ");
             int l,b;
             scanf("%d %d",&l,&b);
             area = l * b;
             printf("\nArea of Rectangle is %.1f",area);
             break;
    case 2 : printf("Enter the TRIANGLE Base and height : ");
             int x,y;
             scanf("%d %d",&x,&y);
             area = (1/2)*x * y;
             printf("\nArea of triangle is %.1f",area);
             break;
    default:
        break;
    }

    return 0;
}
```

15.

```
/*
Problem 4: Error Codes in a Program
Problem Statement:
Write a C program to simulate error handling using enum. The program should:
1.  Define an enum named ErrorCode with values:
o   SUCCESS (0)
o   FILE_NOT_FOUND (1)
o   ACCESS_DENIED (2)
o   OUT_OF_MEMORY (3)
o   UNKNOWN_ERROR (4)
2.  Simulate a function that returns an error code based on a scenario.
3.  Based on the returned error code, print an appropriate message to the
user.


*/
#include <stdio.h>

// Define the ErrorCode enum
enum ErrorCode {
    SUCCESS,          // 0
    FILE_NOT_FOUND,   // 1
    ACCESS_DENIED,    // 2
    OUT_OF_MEMORY,    // 3
    UNKNOWN_ERROR     // 4
};

// Simulate a function that returns an error code
enum ErrorCode simulateErrorScenario(int scenario) {
    switch (scenario) {
        case 0:
            return SUCCESS;
        case 1:
            return FILE_NOT_FOUND;
        case 2:
            return ACCESS_DENIED;
        case 3:
            return OUT_OF_MEMORY;
```

```c
        default:
            return UNKNOWN_ERROR;
    }
}

int main() {
    int scenario;

    // Prompt the user to select a scenario
    printf("Select a scenario (0 for SUCCESS, 1 for FILE_NOT_FOUND, 2 for
ACCESS_DENIED, 3 for OUT_OF_MEMORY, others for UNKNOWN_ERROR): ");
    scanf("%d", &scenario);

    // Get the error code
    enum ErrorCode errorCode = simulateErrorScenario(scenario);

    // Handle the error code and print the appropriate message
    switch (errorCode) {
        case SUCCESS:
            printf("Operation completed successfully.\n");
            break;
        case FILE_NOT_FOUND:
            printf("Error: File not found.\n");
            break;
        case ACCESS_DENIED:
            printf("Error: Access denied.\n");
            break;
        case OUT_OF_MEMORY:
            printf("Error: Out of memory.\n");
            break;
        case UNKNOWN_ERROR:
            printf("Error: Unknown error occurred.\n");
            break;
        default:
            // This case should not be reached
            break;
    }

    return 0;
}
```

16.

```c
/*
Problem 5: User Roles in a System
Problem Statement:
```

```c
Write a C program to define user roles in a system using enum. The program
should:
1.  Define an enum named UserRole with values ADMIN, EDITOR, VIEWER, and
GUEST.
2.  Accept the user role as input (0 for ADMIN, 1 for EDITOR, etc.).
3.  Display the permissions associated with each role:
o   ADMIN: "Full access to the system."
o   EDITOR: "Can edit content but not manage users."
o   VIEWER: "Can view content only."
o   GUEST: "Limited access, view public content only."



*/
#include <stdio.h>

// Define the UserRole enum
enum UserRole {
    ADMIN,   // 0
    EDITOR,  // 1
    VIEWER,  // 2
    GUEST    // 3
};

int main() {
    int input;

    // Prompt the user to select a role
    printf("Enter the user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3
for GUEST): ");
    scanf("%d", &input);

    // Validate the input
    if (input < 0 || input > 3) {
        printf("Invalid input! Please enter 0, 1, 2, or 3.\n");
        return 1; // Exit with error
    }

    // Cast the input to UserRole enum
    enum UserRole role = (enum UserRole)input;

    // Display the permissions based on the role
    switch (role) {
        case ADMIN:
            printf("ADMIN: Full access to the system.\n");
            break;
        case EDITOR:
            printf("EDITOR: Can edit content but not manage users.\n");
            break;
        case VIEWER:
            printf("VIEWER: Can view content only.\n");
            break;
        case GUEST:
            printf("GUEST: Limited access, view public content only.\n");
            break;
        default:
            // This case should never be reached due to input validation
            break;
    }

    return 0; // Exit successfully
}
```

17.

```
#include <stdio.h>
//forced allignment
struct t1{
    int d : 5;
    int m : 4;
    int year;
};
int main(){

    struct t1 test = { 25, 11, 2024};

    printf("date = %d - %d - %d\n",test.d,test.m,test.year);

    return 0;
}
```

18.

```
#include <stdio.h>
//forced allignment
struct t1{
    unsigned int d : 5;
    unsigned int m : 4;
    int year;
};
int main(){

    struct t1 test = { 25, 11, 2024};
```

```c
    struct t1 *ptr = &test;

    printf("date = %d - %d - %d\n",test.d,test.m,test.year);

    return 0;
}
```