In [ ]:

```
# prompt: Write a program in C to store and display information about 5 cars(n
# price) using an array of structures.

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

#define MAX_CARS 5

// Structure to represent a car
struct Car {
    char name[50];
    int year;
    float price;
};

int main() {
    struct Car cars[MAX_CARS];

    // Input car information
    for (int i = 0; i < MAX_CARS; i++) {
        printf("Enter details for car %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]s", cars[i].name); // Read name with spaces
        printf("Year of launch: ");
        scanf("%d", &cars[i].year);
        printf("Price: ");
        scanf("%f", &cars[i].price);
        printf("\n");
    }

    // Display car information
    printf("Car Information:\n");
    for (int i = 0; i < MAX_CARS; i++) {
        printf("Car %d:\n", i + 1);
        printf("Name: %s\n", cars[i].name);
        printf("Year of launch: %d\n", cars[i].year);
        printf("Price: %.2f\n", cars[i].price);
        printf("\n");
    }

    return 0;
}
```

In [ ]:

```
# prompt: Define a structure named Book with members title (string), author (s
# (integer).

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

#define MAX_CARS 5

// Structure to represent a car
struct Car {
```

```c
    char name[50];
    int year;
    float price;
};

// Structure to represent a book
struct Book {
    char title[100];
    char author[50];
    int pages;
};

int main() {
    struct Car cars[MAX_CARS];

    // Input car information
    for (int i = 0; i < MAX_CARS; i++) {
        printf("Enter details for car %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]s", cars[i].name); // Read name with spaces
        printf("Year of launch: ");
        scanf("%d", &cars[i].year);
        printf("Price: ");
        scanf("%f", &cars[i].price);
        printf("\n");
    }

    // Display car information
    printf("Car Information:\n");
    for (int i = 0; i < MAX_CARS; i++) {
        printf("Car %d:\n", i + 1);
        printf("Name: %s\n", cars[i].name);
        printf("Year of launch: %d\n", cars[i].year);
        printf("Price: %.2f\n", cars[i].price);
        printf("\n");
    }

    // Example usage of the Book structure
    struct Book myBook;
    printf("Enter book details:\n");
    printf("Title: ");
    scanf(" %[^\n]s", myBook.title);
    printf("Author: ");
    scanf(" %[^\n]s", myBook.author);
    printf("Pages: ");
    scanf("%d", &myBook.pages);

    printf("\nBook Information:\n");
    printf("Title: %s\n", myBook.title);
    printf("Author: %s\n", myBook.author);
    printf("Pages: %d\n", myBook.pages);
    return 0;
}
```

In [ ]:

```
# prompt: Write a recursive function that checks if a string is a palindrome.
# Accept the string as input.
# Compare the first and last characters of the string, then recursively check
# without the first and last characters.
# Return true if the string is a palindrome, and false otherwise.
# Test the function with multiple strings (both palindromes and non-palindrome

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

bool isPalindrome(char *str, int start, int end) {
    if (start >= end) {
        return true;
    }
    if (str[start] != str[end]) {
        return false;
    }
    return isPalindrome(str, start + 1, end - 1);
}

int main() {
    char str1[] = "racecar";
    char str2[] = "hello";
    char str3[] = "madam";
    char str4[] = "rotor";
    char str5[] = "A man, a plan, a canal: Panama"; // Example with non-alphar


    printf("\"%s\" is a palindrome: %s\n", str1, isPalindrome(str1, 0, strlen(
    printf("\"%s\" is a palindrome: %s\n", str2, isPalindrome(str2, 0, strlen(
    printf("\"%s\" is a palindrome: %s\n", str3, isPalindrome(str3, 0, strlen(
    printf("\"%s\" is a palindrome: %s\n", str4, isPalindrome(str4, 0, strlen(

    //Removing non-alphanumeric characters for accurate palindrome check of st
    char cleanedStr5[100];
    int j = 0;
    for(int i = 0; i < strlen(str5); i++){
        if(isalnum(str5[i])){
            cleanedStr5[j++] = tolower(str5[i]);
        }
    }
    cleanedStr5[j] = '\0';


    printf("\"%s\" is a palindrome: %s\n", str5, isPalindrome(cleanedStr5, 0,

    return 0;
}
```

```
In [ ]: # prompt: Write a function in C to compute the distance between two points
        # and use it to develop another function that will compute the
        # area of the triangle whose vertices are A(x1, y1), B(x2, y2),
        # and C(x3, y3). Use these functions to develop a function
        # which returns a value 1 if the point (x, y) lines inside the
        # triangle ABC, otherwise a value 0.

        #include <stdio.h>
        #include <math.h>

        // Function to compute the distance between two points
        float distance(float x1, float y1, float x2, float y2) {
            return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
        }

        // Function to compute the area of a triangle
        float triangle_area(float x1, float y1, float x2, float y2, float x3, float y3
            return 0.5 * fabs((x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)));
        }

        // Function to check if a point lies inside a triangle
        int is_inside_triangle(float x1, float y1, float x2, float y2, float x3, float
            float area_abc = triangle_area(x1, y1, x2, y2, x3, y3);
            float area_abp = triangle_area(x1, y1, x2, y2, x, y);
            float area_bcp = triangle_area(x2, y2, x3, y3, x, y);
            float area_cap = triangle_area(x3, y3, x1, y1, x, y);

            // Check if the sum of the areas of the three smaller triangles is equal t
            return (fabs(area_abc - (area_abp + area_bcp + area_cap)) < 0.0001) ? 1 :
        }

        int main() {
            float x1, y1, x2, y2, x3, y3, x, y;

            // Input triangle vertices
            printf("Enter the coordinates of the triangle vertices:\n");
            printf("A(x1, y1): ");
            scanf("%f %f", &x1, &y1);
            printf("B(x2, y2): ");
            scanf("%f %f", &x2, &y2);
            printf("C(x3, y3): ");
            scanf("%f %f", &x3, &y3);

            // Input point to check
            printf("Enter the coordinates of the point to check: ");
            scanf("%f %f", &x, &y);


            // Check if the point lies inside the triangle
            if (is_inside_triangle(x1, y1, x2, y2, x3, y3, x, y)) {
                printf("The point (%.2f, %.2f) lies inside the triangle.\n", x, y);
            } else {
                printf("The point (%.2f, %.2f) lies outside the triangle.\n", x, y);
```

```
    }
    return 0;
}
```

In [ ]:

```
# prompt: Write a program in Cto store and display information about multiple
# price) using an array of structures

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

#define MAX_BOOKS 100 // Adjust as needed

// Structure to represent a book
struct Book {
    char title[100];
    char author[50];
    float price;
};

int main() {
    struct Book books[MAX_BOOKS];
    int numBooks = 0;

    // Input book information
    char choice;
    do {
        printf("Enter details for book %d:\n", numBooks + 1);
        printf("Title: ");
        scanf(" %[^\n]s", books[numBooks].title); // Read name with spaces
        printf("Author: ");
        scanf(" %[^\n]s", books[numBooks].author);
        printf("Price: ");
        scanf("%f", &books[numBooks].price);
        numBooks++;

        printf("Do you want to enter another book? (y/n): ");
        scanf(" %c", &choice);
        printf("\n");
    } while (choice == 'y' || choice == 'Y' && numBooks < MAX_BOOKS);


    // Display book information
    printf("Book Information:\n");
    for (int i = 0; i < numBooks; i++) {
        printf("Book %d:\n", i + 1);
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Price: %.2f\n", books[i].price);
        printf("\n");
    }

    return 0;
}
```

In [ ]:
```
# prompt: Write a C Program to Add Two Complex Numbers by Passing Structure to

#include <stdio.h>

// Structure to represent a complex number
struct Complex {
    float real;
    float imag;
};

// Function to add two complex numbers
struct Complex addComplex(struct Complex num1, struct Complex num2) {
    struct Complex sum;
    sum.real = num1.real + num2.real;
    sum.imag = num1.imag + num2.imag;
    return sum;
}

int main() {
    struct Complex num1, num2, sum;

    // Input the first complex number
    printf("Enter the real and imaginary parts of the first complex number:\n"
    scanf("%f %f", &num1.real, &num1.imag);

    // Input the second complex number
    printf("Enter the real and imaginary parts of the second complex number:\r
    scanf("%f %f", &num2.real, &num2.imag);

    // Add the complex numbers using the function
    sum = addComplex(num1, num2);

    // Display the sum
    printf("Sum of the complex numbers: %.2f + %.2fi\n", sum.real, sum.imag);

    return 0;
}
```

In [ ]:
```
# prompt: Write a program in C to define a structure named Employee with field
# salary. Accept details of 5 employees from the user and sort them by name ir
# order using a function. Display the sorted list.

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

#define MAX_EMPLOYEES 5

// Structure to represent an employee
struct Employee {
    int id;
    char name[50];
    float salary;
};
```

```
// Function to compare two employee names for sorting
int compareEmployees(const void *a, const void *b) {
    return strcmp(((struct Employee *)a)->name, ((struct Employee *)b)->name);
}

int main() {
    struct Employee employees[MAX_EMPLOYEES];

    // Input employee details
    for (int i = 0; i < MAX_EMPLOYEES; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("ID: ");
        scanf("%d", &employees[i].id);
        printf("Name: ");
        scanf(" %[^\n]s", employees[i].name); // Read name with spaces
        printf("Salary: ");
        scanf("%f", &employees[i].salary);
        printf("\n");
    }

    // Sort the employees by name
    qsort(employees, MAX_EMPLOYEES, sizeof(struct Employee), compareEmployees)

    // Display the sorted employee list
    printf("Sorted Employee List:\n");
    for (int i = 0; i < MAX_EMPLOYEES; i++) {
        printf("Employee %d:\n", i + 1);
        printf("ID: %d\n", employees[i].id);
        printf("Name: %s\n", employees[i].name);
        printf("Salary: %.2f\n", employees[i].salary);
        printf("\n");
    }

    return 0;
}
```

In [ ]:

```
# prompt: Write a C program that defines a function to swap two arrays of the
# using any extra memory or a third array. The function should swap the elemen
# arrays in place. After calling the function, print the modified arrays. Usin

#include <stdio.h>

void swap_arrays(int arr1[], int arr2[], int size) {
    for (int i = 0; i < size; i++) {
        arr1[i] = arr1[i] ^ arr2[i];
        arr2[i] = arr1[i] ^ arr2[i];
        arr1[i] = arr1[i] ^ arr2[i];
    }
}

int main() {
    int size;
    printf("Enter the size of the arrays: ");
    scanf("%d", &size);

    int arr1[size], arr2[size];

    printf("Enter elements of the first array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr1[i]);
    }

    printf("Enter elements of the second array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr2[i]);
    }

    swap_arrays(arr1, arr2, size);

    printf("Modified first array:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr1[i]);
    }
    printf("\n");

    printf("Modified second array:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr2[i]);
    }
    printf("\n");

    return 0;
}
```

In [ ]:

```
# prompt: Write a program that converts a string like "124" to an integer
# 124.

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

int stringToInt(const char *str) {
    int result = 0;
    int sign = 1;
    int i = 0;

    // Handle optional leading sign
    if (str[0] == '-') {
        sign = -1;
        i++;
    } else if (str[0] == '+') {
        i++;
    }
    // Convert digits to integer
    for (; str[i] != '\0'; i++) {
        if (str[i] >= '0' && str[i] <= '9') {
            result = result * 10 + (str[i] - '0');
        } else {
            // Handle non-digit characters (optional error handling)
            fprintf(stderr, "Invalid character in string: %c\n", str[i]);
            return 0; // or handle the error differently
        }
    }

    return sign * result;
}

int main() {
    char str[] = "124";
    int num = stringToInt(str);
    printf("%d\n", num);
    return 0;
}
```

In [ ]:
```
# prompt: Write a recursive function to find all prime numbers up to a given n

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

// Function to check if a number is prime
bool is_prime(int n) {
    if (n <= 1) {
        return false;
    }
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

// Recursive function to find all prime numbers up to a given number
void find_primes(int n, int current) {
    if (current > n) {
        return;
    }

    if (is_prime(current)) {
        printf("%d ", current);
    }

    find_primes(n, current + 1);
}

int main() {
    int n;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    printf("Prime numbers up to %d are: ", n);
    find_primes(n, 2); // Start checking from 2
    printf("\n");

    return 0;
}
```

In [ ]:

```
# prompt: Write a program in C to define a structure named Employee with fielc
# salary. Accept details of 5 employees from the user and sort them by salary
# order using a function. Display the sorted list.

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

#define MAX_EMPLOYEES 5

// Structure to represent an employee
struct Employee {
    int id;
    char name[50];
    float salary;
};

// Function to compare two employees based on salary for sorting
int compareEmployees(const void *a, const void *b) {
    struct Employee *empA = (struct Employee *)a;
    struct Employee *empB = (struct Employee *)b;
    if (empA->salary < empB->salary) {
        return -1;
    } else if (empA->salary > empB->salary) {
        return 1;
    } else {
        return 0;
    }
}

int main() {
    struct Employee employees[MAX_EMPLOYEES];

    // Input employee details
    for (int i = 0; i < MAX_EMPLOYEES; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("ID: ");
        scanf("%d", &employees[i].id);
        printf("Name: ");
        scanf(" %[^\n]s", employees[i].name); // Read name with spaces
        printf("Salary: ");
        scanf("%f", &employees[i].salary);
        printf("\n");
    }

    // Sort the employees by salary using qsort
    qsort(employees, MAX_EMPLOYEES, sizeof(struct Employee), compareEmployees)

    // Display the sorted employee list
    printf("Sorted Employee List (by salary):\n");
    for (int i = 0; i < MAX_EMPLOYEES; i++) {
        printf("Employee %d:\n", i + 1);
        printf("ID: %d\n", employees[i].id);
        printf("Name: %s\n", employees[i].name);
        printf("Salary: %.2f\n", employees[i].salary);
        printf("\n");
    }

    return 0;
}
```

In [ ]:
```
# prompt: Write a program using a function to calculate the hypotenuse of a ri

#include <stdio.h>
#include <math.h>

// Function to calculate the hypotenuse
float calculateHypotenuse(float a, float b) {
    return sqrt(a * a + b * b);
}

int main() {
    float sideA, sideB, hypotenuse;

    // Get input from the user
    printf("Enter the length of side A: ");
    scanf("%f", &sideA);
    printf("Enter the length of side B: ");
    scanf("%f", &sideB);

    // Calculate the hypotenuse
    hypotenuse = calculateHypotenuse(sideA, sideB);

    // Display the result
    printf("The length of the hypotenuse is: %.2f\n", hypotenuse);

    return 0;
}
```

In [ ]:
```
# prompt: Write a program in C to find the sum of the series 1!/1+2!/2+3!/3+4!
# function.

#include <stdio.h>
```

```
// Function to calculate factorial
long long factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        long long fact = 1;
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }
}

int main() {
    float sum = 0;
    for (int i = 1; i <= 5; i++) {
        sum += (float)factorial(i) / i;
    }
    printf("Sum of the series: %.2f\n", sum);
    return 0;
}
```

In [ ]:
```
# prompt: Design and implement a C program to manage student information using
# Requirements:
# Define a structure named Student to represent a student. The structure shoul
# following members:
# roll_no: An integer to store the student's roll number.
# name: A character array to store the student's name.
# marks: An array of integers to store the marks obtained in five subjects (e.
# English, History, Geography).

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

#define MAX_STUDENTS 100 // Adjust as needed
#define MAX_SUBJECTS 5

// Structure to represent a student
struct Student {
    int roll_no;
    char name[50];
    int marks[MAX_SUBJECTS];
};

int main() {
    struct Student students[MAX_STUDENTS];
    int numStudents = 0;
    char choice;

    do {
        printf("Enter details for student %d:\n", numStudents + 1);
        printf("Roll number: ");
        scanf("%d", &students[numStudents].roll_no);
        printf("Name: ");
        scanf(" %[^\n]s", students[numStudents].name); // Read name with space
        printf("Marks in 5 subjects:\n");
        for (int i = 0; i < MAX_SUBJECTS; i++) {
            printf("Subject %d: ", i + 1);
            scanf("%d", &students[numStudents].marks[i]);
        }
        numStudents++;

        printf("Do you want to enter another student? (y/n): ");
        scanf(" %c", &choice);
        printf("\n");

    } while (choice == 'y' || choice == 'Y' && numStudents < MAX_STUDENTS);

    // Display student information
    printf("Student Information:\n");
    for (int i = 0; i < numStudents; i++) {
        printf("Student %d:\n", i + 1);
        printf("Roll Number: %d\n", students[i].roll_no);
        printf("Name: %s\n", students[i].name);
        printf("Marks:\n");
        for (int j = 0; j < MAX_SUBJECTS; j++) {
            printf("Subject %d: %d\n", j + 1, students[i].marks[j]);
        }
        printf("\n");
    }

    return 0;
}
```

In [ ]:

```
# prompt: There is a structure called employee that holds information like emp
# date of joining. Write a program to create an array of structures and enter
# Then ask the user to enter current date. Display the names of those employee
# tenure is greater than equal to 3 years.

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

#define MAX_EMPLOYEES 100

// Structure to represent an employee
struct Employee {
    int code;
    char name[50];
    int year;
    int month;
    int day;
};

int main() {
    struct Employee employees[MAX_EMPLOYEES];
    int numEmployees = 0;
    char choice;

    // Input employee details
    do {
        printf("Enter details for employee %d:\n", numEmployees + 1);
        printf("Code: ");
        scanf("%d", &employees[numEmployees].code);
        printf("Name: ");
        scanf(" %[^\n]s", employees[numEmployees].name);
        printf("Date of Joining (YYYY MM DD): ");
        scanf("%d %d %d", &employees[numEmployees].year, &employees[numEmploye
        numEmployees++;

        printf("Do you want to enter another employee? (y/n): ");
        scanf(" %c", &choice);
        printf("\n");
    } while (choice == 'y' || choice == 'Y' && numEmployees < MAX_EMPLOYEES);

    // Input current date
    int currentYear, currentMonth, currentDay;
    printf("Enter current date (YYYY MM DD): ");
    scanf("%d %d %d", &currentYear, &currentMonth, &currentDay);

    // Calculate and display employees with tenure >= 3 years
    printf("Employees with tenure >= 3 years:\n");
    for (int i = 0; i < numEmployees; i++) {
        int tenureYears = currentYear - employees[i].year;

        if (currentMonth < employees[i].month || (currentMonth == employees[i]
          tenureYears--;
        }

        if (tenureYears >= 3) {
            printf("Name: %s\n", employees[i].name);
        }
    }

    return 0;
}
```

In [ ]:

```
# prompt: Write a C program to find a factorial of a number using user-definec

#include <stdio.h>

// Function to calculate factorial
long long factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        long long fact = 1;
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }
}

int main() {
    int num;

    printf("Enter a non-negative integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        long long fact = factorial(num);
        printf("Factorial of %d = %lld\n", num, fact);
    }

    return 0;
}
```

In [ ]:

```
# prompt: Design a program to store and display a Date using a structure with
# and year. In the same program, use a union to store either a timestamp (as a
# formatted_date (as a string). Provide functions to input and display both fo
# date.

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

// Structure for Date
struct Date {
    int day;
    int month;
    int year;
};

// Union to store either timestamp or formatted date
union DateRepresentation {
    time_t timestamp;
    char formatted_date[20]; // Adjust size as needed
};

// Function to input date
void inputDate(struct Date *date) {
    printf("Enter day: ");
    scanf("%d", &date->day);
    printf("Enter month: ");
    scanf("%d", &date->month);
    printf("Enter year: ");
    scanf("%d", &date->year);
}
```

```c
// Function to display date from structure
void displayDate(struct Date date) {
    printf("Date: %d/%d/%d\n", date.day, date.month, date.year);
}

// Function to input date as timestamp
void inputTimestamp(union DateRepresentation *dateRep) {
    printf("Enter timestamp (seconds since epoch): ");
    scanf("%ld", &dateRep->timestamp);
}

// Function to display date as timestamp
void displayTimestamp(union DateRepresentation dateRep) {
    printf("Timestamp: %ld\n", dateRep.timestamp);
    struct tm *timeinfo = localtime(&dateRep.timestamp);
    char buffer[20];
    strftime(buffer, 20, "%Y-%m-%d", timeinfo);
    printf("Formatted Date: %s\n", buffer);
}

// Function to input formatted date
void inputFormattedDate(union DateRepresentation *dateRep) {
    printf("Enter formatted date (YYYY-MM-DD): ");
    scanf("%s", dateRep->formatted_date);
}


// Function to display formatted date
void displayFormattedDate(union DateRepresentation dateRep){
    printf("Formatted Date: %s\n", dateRep.formatted_date);
}

int main() {
    struct Date date;
    union DateRepresentation dateRep;

    // Input and display date using structure
    printf("Enter date using structure:\n");
    inputDate(&date);
    displayDate(date);

    // Input and display date as timestamp
    printf("\nEnter date as timestamp:\n");
    inputTimestamp(&dateRep);
    displayTimestamp(dateRep);

    // Input and display formatted date
    printf("\nEnter date as formatted date:\n");
    inputFormattedDate(&dateRep);
    displayFormattedDate(dateRep);

    return 0;
}
```

```
In [ ]: # prompt: Define a union Shape containing structures for Circle, Rectangle, an
        # parameters.
        # Write a function to calculate and return the area based on user selection in

        #include <stdio.h>
        #include <math.h>

        // Define structures for Circle, Rectangle, and Triangle
        typedef struct {
            float radius;
        } Circle;

        typedef struct {
            float length;
            float width;
        } Rectangle;

        typedef struct {
            float base;
            float height;
        } Triangle;

        // Union to represent different shapes
        typedef union {
            Circle circle;
            Rectangle rectangle;
            Triangle triangle;
        } Shape;

        // Function to calculate the area of a shape
        float calculateArea(Shape shape, int choice) {
            float area = 0.0;
            switch (choice) {
                case 1: // Circle
                    area = M_PI * shape.circle.radius * shape.circle.radius;
                    break;
                case 2: // Rectangle
                    area = shape.rectangle.length * shape.rectangle.width;
                    break;
                case 3: // Triangle
                    area = 0.5 * shape.triangle.base * shape.triangle.height;
                    break;
                default:
                    printf("Invalid choice!\n");
                    break;
            }
            return area;
        }

        int main() {
            Shape shape;
            int choice;
```

```
    // Get user input for the shape type
    printf("Select the shape:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("3. Triangle\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    // Get shape parameters
    switch (choice) {
        case 1: // Circle
            printf("Enter the radius of the circle: ");
            scanf("%f", &shape.circle.radius);
            break;
        case 2: // Rectangle
            printf("Enter the length of the rectangle: ");
            scanf("%f", &shape.rectangle.length);
            printf("Enter the width of the rectangle: ");
            scanf("%f", &shape.rectangle.width);
            break;
        case 3: // Triangle
            printf("Enter the base of the triangle: ");
            scanf("%f", &shape.triangle.base);
            printf("Enter the height of the triangle: ");
            scanf("%f", &shape.triangle.height);
            break;
        default:
            printf("Invalid choice!\n");
            return 1;
    }

    // Calculate and print the area
    float area = calculateArea(shape, choice);
    printf("The area of the selected shape is: %.2f\n", area);

    return 0;
}
```

In [ ]:
```
# prompt: Design a structure named "Car" to store details like car ID, model,
# Write a C program to input data for three cars, calculate the total rental c
# number of days, and display the results.

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

#define MAX_CARS 3

// Structure to store car details
struct Car {
    int car_id;
    char model[50];
    float rental_rate;
};
```

```
int main() {
    struct Car cars[MAX_CARS];
    int num_days;

    // Input data for three cars
    for (int i = 0; i < MAX_CARS; i++) {
        printf("Enter details for car %d:\n", i + 1);
        printf("Car ID: ");
        scanf("%d", &cars[i].car_id);
        printf("Model: ");
        scanf(" %[^\n]s", cars[i].model); // Read the entire line for model na
        printf("Rental rate per day: ");
        scanf("%f", &cars[i].rental_rate);
        printf("\n");
    }

    // Input the number of days for rental
    printf("Enter the number of days for rental: ");
    scanf("%d", &num_days);

    // Calculate and display the total rental cost for each car
    printf("\nTotal rental cost for each car:\n");
    for (int i = 0; i < MAX_CARS; i++) {
        float total_cost = cars[i].rental_rate * num_days;
        printf("Car ID: %d, Model: %s, Total Cost: %.2f\n", cars[i].car_id, ca
    }

    return 0;
}
```

In [ ]:

```
# prompt: Create a structure Patient with details: name, age, disease, admissi
# bill_amount.
# Implement functions to:
# Accept and display patient records.
# List all patients suffering from a specific disease.
# Calculate and print the total revenue generated by all patients.

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

#define MAX_PATIENTS 100

// Structure to represent a patient
struct Patient {
    char name[50];
    int age;
    char disease[50];
    struct tm admission_date;
    float bill_amount;
};
```

```
// Function to accept patient records
void acceptPatientRecords(struct Patient patients[], int *numPatients) {
    printf("Enter patient details:\n");
    for (int i = *numPatients; i < MAX_PATIENTS; i++) {
        printf("Patient %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]s", patients[i].name);
        printf("Age: ");
        scanf("%d", &patients[i].age);
        printf("Disease: ");
        scanf(" %[^\n]s", patients[i].disease);
        printf("Admission Date (YYYY MM DD): ");
        scanf("%d %d %d", &patients[i].admission_date.tm_year, &patients[i].ac
        patients[i].admission_date.tm_year -= 1900; // Adjust year for struct
        patients[i].admission_date.tm_mon -= 1;   // Adjust month for struct t
        printf("Bill Amount: ");
        scanf("%f", &patients[i].bill_amount);

        (*numPatients)++;
        char another;
        printf("Add another patient? (y/n): ");
        scanf(" %c", &another);
        if (another != 'y') {
            break;
        }
    }
}

// Function to display patient records
void displayPatientRecords(struct Patient patients[], int numPatients) {
    printf("\nPatient Records:\n");
    for (int i = 0; i < numPatients; i++) {
        printf("Patient %d:\n", i + 1);
        printf("Name: %s\n", patients[i].name);
        printf("Age: %d\n", patients[i].age);
        printf("Disease: %s\n", patients[i].disease);
        printf("Admission Date: %d-%d-%d\n", patients[i].admission_date.tm_yea
        printf("Bill Amount: %.2f\n\n", patients[i].bill_amount);
    }
}

// Function to list patients with a specific disease
void listPatientsWithDisease(struct Patient patients[], int numPatients, char
    printf("Patients with %s:\n", disease);
    for (int i = 0; i < numPatients; i++) {
        if (strcmp(patients[i].disease, disease) == 0) {
            printf("Name: %s, Age: %d\n", patients[i].name, patients[i].age);
        }
    }
}

// Function to calculate total revenue
```

```
float calculateTotalRevenue(struct Patient patients[], int numPatients) {
    float totalRevenue = 0;
    for (int i = 0; i < numPatients; i++) {
        totalRevenue += patients[i].bill_amount;
    }
    return totalRevenue;
}

int main() {
    struct Patient patients[MAX_PATIENTS];
    int numPatients = 0;

    acceptPatientRecords(patients, &numPatients);
    displayPatientRecords(patients, numPatients);

    char targetDisease[50];
    printf("Enter disease to search for: ");
    scanf(" %[^\n]s", targetDisease);
    listPatientsWithDisease(patients, numPatients, targetDisease);

    float totalRevenue = calculateTotalRevenue(patients, numPatients);
    printf("Total Revenue: %.2f\n", totalRevenue);

    return 0;
}
```

In [ ]:

```
# prompt: Write a program in C to compare two strings without using string lib

#include <stdio.h>

int main() {
    char str1[100], str2[100];
    int i = 0, flag = 0;

    printf("Enter the first string: ");
    scanf("%s", str1);
    printf("Enter the second string: ");
    scanf("%s", str2);

    while (str1[i] != '\0' || str2[i] != '\0') {
        if (str1[i] != str2[i]) {
            flag = 1;
            break;
        }
        i++;
    }

    if (flag == 0) {
        printf("Strings are equal.\n");
    } else {
        printf("Strings are not equal.\n");
    }
    return 0;
}
```

In [ ]:
```
# prompt: Write a C program to input a 3×3 matrix from the user and compute th
# elements. Display the entered matrix and the calculated sum.

#include <stdio.h>

int main() {
    int matrix[3][3];
    int sum = 0;

    // Input matrix elements
    printf("Enter the elements of the 3x3 matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display the entered matrix
    printf("Entered matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }

    // Compute the sum of diagonal elements
    for (int i = 0; i < 3; i++) {
        sum += matrix[i][i];
    }

    // Display the sum
    printf("Sum of diagonal elements: %d\n", sum);

    return 0;
}
```

In [ ]:
```
# prompt: Define a union ExamScore containing fields for three different types
# (out of 100), grade (A/B/C/D), and percentage.
# Implement a program that takes student input and selects the appropriate for
# storing scores based on the type of exam.

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

// Define the union ExamScore
union ExamScore {
    int marks_100;
    char grade;
```

```c
    float percentage;
};

int main() {
    union ExamScore score;
    int examType;
    char anotherStudent;

    do {
        printf("Enter the type of exam:\n");
        printf("1. Marks out of 100\n");
        printf("2. Grade (A/B/C/D)\n");
        printf("3. Percentage\n");
        printf("Enter your choice: ");
        scanf("%d", &examType);

        switch (examType) {
            case 1:
                printf("Enter marks out of 100: ");
                scanf("%d", &score.marks_100);
                printf("Marks: %d\n", score.marks_100);
                break;
            case 2:
                printf("Enter grade (A/B/C/D): ");
                scanf(" %c", &score.grade); // Note the space before %c to cor
                printf("Grade: %c\n", score.grade);
                break;
            case 3:
                printf("Enter percentage: ");
                scanf("%f", &score.percentage);
                printf("Percentage: %.2f\n", score.percentage);
                break;
            default:
                printf("Invalid exam type.\n");
        }

        printf("Enter another student's details? (y/n): ");
        scanf(" %c", &anotherStudent); // Note the space before %c

    } while (anotherStudent == 'y' || anotherStudent == 'Y');

    return 0;
}
```

In [ ]:

```
# prompt: Write a program for a simple library management system using a struc
# should store:
# Book Title (string).
# Book ID (integer).
# Book Details (using a union to handle two types of books):
# Regular Book (with price and number of pages).
# E-Book (with file size and format).
# The program should allow the user to select a book type and input its detail
# the details based on the selected book type.
```

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

#define MAX_BOOKS 100

// Structure for regular book details
typedef struct {
    float price;
    int num_pages;
} RegularBook;

// Structure for e-book details
typedef struct {
    float file_size;
    char format[20];
} EBook;

// Union to store either regular book or e-book details
typedef union {
    RegularBook regular;
    EBook ebook;
} BookDetails;

// Structure to represent a book
typedef struct {
    char title[100];
    int id;
    BookDetails details;
    int book_type; // 1 for regular book, 2 for e-book
} Book;

int main() {
    Book library[MAX_BOOKS];
    int num_books = 0;
    char another_book;

    do {
        printf("Enter book details:\n");
        printf("Book Title: ");
        scanf(" %[^\n]s", library[num_books].title);
        printf("Book ID: ");
        scanf("%d", &library[num_books].id);

        printf("Select book type:\n");
        printf("1. Regular Book\n");
        printf("2. E-Book\n");
        printf("Enter your choice: ");
        scanf("%d", &library[num_books].book_type);


        if (library[num_books].book_type == 1) {
            printf("Enter price: ");
            scanf("%f", &library[num_books].details.regular.price);
            printf("Enter number of pages: ");
            scanf("%d", &library[num_books].details.regular.num_pages);
        } else if (library[num_books].book_type == 2) {
            printf("Enter file size: ");
            scanf("%f", &library[num_books].details.ebook.file_size);
            printf("Enter format: ");
            scanf(" %[^\n]s", library[num_books].details.ebook.format);
        } else {
            printf("Invalid book type.\n");
            continue; // Skip to the next iteration
        }

        num_books++;

        printf("Enter details for another book? (y/n): ");
        scanf(" %c", &another_book);

    } while (another_book == 'y' && num_books < MAX_BOOKS);

    printf("\nLibrary Details:\n");
    for (int i = 0; i < num_books; i++) {
        printf("Book Title: %s\n", library[i].title);
        printf("Book ID: %d\n", library[i].id);

        if (library[i].book_type == 1) {
            printf("Price: %.2f\n", library[i].details.regular.price);
            printf("Number of Pages: %d\n", library[i].details.regular.num_pag
        } else {
            printf("File Size: %.2f\n", library[i].details.ebook.file_size);
            printf("Format: %s\n", library[i].details.ebook.format);
        }
        printf("\n");
    }

    return 0;
}
```

In [ ]:
```
# prompt: Write a C program to find the LCM (Least Common Multiple) of two num
# user-defined function.

#include <stdio.h>

// Function to calculate the LCM of two numbers
int findLCM(int num1, int num2) {
    int max = (num1 > num2) ? num1 : num2;
    while (1) {
        if (max % num1 == 0 && max % num2 == 0) {
            return max;
        }
        max++;
    }
}
```

```
int main() {
    int num1, num2;

    printf("Enter two positive integers: ");
    scanf("%d %d", &num1, &num2);

    if (num1 <= 0 || num2 <= 0) {
        printf("Please enter positive integers.\n");
    } else {
        int lcm = findLCM(num1, num2);
        printf("The LCM of %d and %d is %d.\n", num1, num2, lcm);
    }

    return 0;
}
```

In [ ]:

```
# prompt: Write a C program that takes the temperature in Celsius from the use
# Fahrenheit using the formula: F=(9/5)*C+32 Return the Fahrenheit temperature
# function and display it

#include <stdio.h>

// Function to convert Celsius to Fahrenheit
float celsiusToFahrenheit(float celsius) {
    return (9.0 / 5.0) * celsius + 32.0;
}

int main() {
    float celsius, fahrenheit;

    // Input temperature in Celsius
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);

    // Convert Celsius to Fahrenheit using the function
    fahrenheit = celsiusToFahrenheit(celsius);

    // Display the Fahrenheit temperature
    printf("Temperature in Fahrenheit: %.2f\n", fahrenheit);

    return 0;
}
```

OPTIONAL QUESTIONS

In [ ]:

```
# prompt: Write a program that defines a structure Book with the following mem
# char title[100]
# char author[100]
# float price
# Initialize an array of Book structures with sample data.
# Write a function to display the details of all books.
# Write a function to search for a book by title and return the book's details
# Demonstrate the program with 3 books.

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

#define MAX_BOOKS 3

// Structure to represent a book
typedef struct {
    char title[100];
    char author[100];
    float price;
} Book;

// Function to display book details
void displayBookDetails(Book books[], int numBooks) {
    printf("Book Details:\n");
    for (int i = 0; i < numBooks; i++) {
        printf("Book %d:\n", i + 1);
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Price: %.2f\n\n", books[i].price);
    }
}

// Function to search for a book by title
void searchBookByTitle(Book books[], int numBooks) {
    char searchTitle[100];
    printf("Enter the title to search for: ");
    scanf(" %[^\n]s", searchTitle); // Read the entire line for the title

    int found = 0;
    for (int i = 0; i < numBooks; i++) {
        if (strcmp(books[i].title, searchTitle) == 0) {
            printf("Book found:\n");
            printf("Title: %s\n", books[i].title);
            printf("Author: %s\n", books[i].author);
            printf("Price: %.2f\n", books[i].price);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Book not found.\n");
    }
}

int main() {
    Book books[MAX_BOOKS];

    // Initialize book data
    strcpy(books[0].title, "The Catcher in the Rye");
    strcpy(books[0].author, "J.D. Salinger");
    books[0].price = 10.99;

    strcpy(books[1].title, "To Kill a Mockingbird");
    strcpy(books[1].author, "Harper Lee");
    books[1].price = 8.50;

    strcpy(books[2].title, "1984");
    strcpy(books[2].author, "George Orwell");
    books[2].price = 12.75;

    // Display book details
    displayBookDetails(books, MAX_BOOKS);

    // Search for a book
    searchBookByTitle(books, MAX_BOOKS);

    return 0;
}
```

In [ ]:

```
# prompt: Write a program in C to manage a simple ATM system using a union for
# transaction types like balance inquiry and withdrawal.

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

#define MAX_TRANSACTIONS 100

// Structure for transaction details
typedef struct {
    char transaction_type[20]; // "balance_inquiry", "withdrawal"
    float amount;
    char timestamp[20]; // Formatted date and time
} Transaction;

// Structure to represent a customer account
typedef struct {
    int account_number;
    char name[50];
    float balance;
    Transaction transactions[MAX_TRANSACTIONS];
    int num_transactions;
} Account;

// Union to handle different transaction types
typedef union {
    float withdrawal_amount;
    // Add more fields for other transaction types
} TransactionData;

// Function to perform a transaction
void performTransaction(Account *account, const char *transactionType, Transac
    if (account == NULL) {
        printf("Invalid account.\n");
        return;
    }
    // Record current time
    time_t timer;
    time(&timer);
    char time_buf[26];
    ctime_r(&timer, time_buf);
    time_buf[strlen(time_buf)-1] = '\0';


    if (strcmp(transactionType, "balance_inquiry") == 0) {
        account->transactions[account->num_transactions].amount = 0;
        strcpy(account->transactions[account->num_transactions].transaction_ty
        strcpy(account->transactions[account->num_transactions].timestamp, tim
        account->num_transactions++;

        printf("Account Balance: $%.2f\n", account->balance);
    } else if (strcmp(transactionType, "withdrawal") == 0) {
        if (account->balance >= transactionData.withdrawal_amount && transacti
          account->balance -= transactionData.withdrawal_amount;
          account->transactions[account->num_transactions].amount = transactic
          strcpy(account->transactions[account->num_transactions].transaction_
          strcpy(account->transactions[account->num_transactions].timestamp, t
          account->num_transactions++;
          printf("Withdrawal successful.\n");
          printf("New balance: $%.2f\n", account->balance);
        } else {
            printf("Insufficient funds or invalid withdrawal amount.\n");
        }
    } else {
        printf("Invalid transaction type.\n");
    }
}

int main() {
    Account customer;

    // Initialize the customer account
    printf("Enter account number: ");
    scanf("%d", &customer.account_number);
    printf("Enter account holder name: ");
    scanf(" %[^\n]s", customer.name);
    printf("Enter initial balance: ");
    scanf("%f", &customer.balance);
    customer.num_transactions = 0;

    int choice;
    TransactionData transactionData;

    do {
        printf("\nATM Menu:\n");
        printf("1. Balance Inquiry\n");
        printf("2. Withdrawal\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                performTransaction(&customer, "balance_inquiry", transactionDa
                break;
            case 2:
                printf("Enter withdrawal amount: ");
                scanf("%f", &transactionData.withdrawal_amount);
                performTransaction(&customer, "withdrawal", transactionData);
                break;
            case 3:
                printf("Exiting ATM.\n");
                break;
            default:
                printf("Invalid choice.\n");
        }
    } while (choice != 3);


    printf("\nTransaction history for account %d:\n", customer.account_number)
    for (int i = 0; i < customer.num_transactions; i++) {
        printf("Transaction Type: %s, Amount: $%.2f, Timestamp: %s\n",
               customer.transactions[i].transaction_type,
               customer.transactions[i].amount,
               customer.transactions[i].timestamp);
    }

    return 0;
}
```

In [ ]:

```
# prompt: Define a structure Book with fields title, author, and price. Write
# Add a new book.
# Display book details.
# Calculate the total price of all books.
# Implement these functions and demonstrate their usage in a menu-driven
# program in C

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

#define MAX_BOOKS 100

// Structure to represent a book
typedef struct {
    char title[100];
    char author[100];
    float price;
```

```c
} Book;

Book books[MAX_BOOKS];
int num_books = 0;

// Function to add a new book
void addBook() {
    if (num_books < MAX_BOOKS) {
        printf("Enter book title: ");
        scanf(" %[^\n]s", books[num_books].title);
        printf("Enter author name: ");
        scanf(" %[^\n]s", books[num_books].author);
        printf("Enter book price: ");
        scanf("%f", &books[num_books].price);
        num_books++;
        printf("Book added successfully!\n");
    } else {
        printf("Library is full. Cannot add more books.\n");
    }
}

// Function to display book details
void displayBookDetails() {
    if (num_books == 0) {
        printf("No books in the library.\n");
        return;
    }
    for (int i = 0; i < num_books; i++) {
        printf("Book %d:\n", i + 1);
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Price: %.2f\n\n", books[i].price);
    }
}

// Function to calculate the total price of all books
float calculateTotalPrice() {
    float total_price = 0;
    for (int i = 0; i < num_books; i++) {
        total_price += books[i].price;
    }
    return total_price;
}

int main() {
    int choice;

    do {
        printf("\nLibrary Menu:\n");
        printf("1. Add a new book\n");
        printf("2. Display book details\n");
        printf("3. Calculate total price of all books\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addBook();
                break;
            case 2:
                displayBookDetails();
                break;
            case 3:
                printf("Total price of all books: %.2f\n", calculateTotalPrice
                break;
            case 4:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

In [ ]:

```
# prompt: Write a program in C that replaces two or more consecutive blanks
# in a string by a single blank. For example, if the input is
# Grim return to the planet of apes!!
# the output should be
# Grim return to the planet of apes!!

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

int main() {
    char str[100];
    char result[100];
    int i = 0, j = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin); // Use fgets to handle spaces in the input

    // Remove trailing newline character if present
    str[strcspn(str, "\n")] = 0;


    // Iterate through the string
    for (i = 0; str[i] != '\0'; i++) {
        // If the current character is a blank
        if (str[i] == ' ') {
            // Check if it's the first blank or if the previous character was
            if (j == 0 || result[j - 1] != ' ') {
                result[j++] = str[i];
            }
        } else {
            result[j++] = str[i];
        }
    }
    result[j] = '\0'; // Null-terminate the result string

    printf("Modified string: %s\n", result);

    return 0;
}
```

In [ ]:

```
# prompt: Define a structure Complex with fields real and imaginary.
# Write functions to:
# Add two complex numbers.
# Multiply two complex numbers.
# Find the magnitude of a complex number.

#include <stdio.h>
#include <math.h>

// Define the structure Complex
typedef struct {
    float real;
    float imaginary;
} Complex;

// Function to add two complex numbers
Complex addComplex(Complex num1, Complex num2) {
    Complex result;
    result.real = num1.real + num2.real;
    result.imaginary = num1.imaginary + num2.imaginary;
    return result;
}

// Function to multiply two complex numbers
Complex multiplyComplex(Complex num1, Complex num2) {
    Complex result;
    result.real = (num1.real * num2.real) - (num1.imaginary * num2.imaginary);
    result.imaginary = (num1.real * num2.imaginary) + (num1.imaginary * num2.r
    return result;
}

// Function to find the magnitude of a complex number
float magnitudeComplex(Complex num) {
    return sqrt((num.real * num.real) + (num.imaginary * num.imaginary));
}

int main() {
    Complex num1, num2, sum, product;
    float magnitude;

    // Input the first complex number
    printf("Enter the real and imaginary parts of the first complex number: ")
    scanf("%f %f", &num1.real, &num1.imaginary);

    // Input the second complex number
    printf("Enter the real and imaginary parts of the second complex number: "
    scanf("%f %f", &num2.real, &num2.imaginary);

    // Add the complex numbers
    sum = addComplex(num1, num2);
    printf("Sum: %.2f + %.2fi\n", sum.real, sum.imaginary);

    // Multiply the complex numbers
    product = multiplyComplex(num1, num2);
    printf("Product: %.2f + %.2fi\n", product.real, product.imaginary);

    // Find the magnitude of the first complex number
    magnitude = magnitudeComplex(num1);
    printf("Magnitude of the first complex number: %.2f\n", magnitude);

    return 0;
}
```

In [ ]:

```
# prompt: Write a C program to find the GCD (Greatest Common Divisor) of two n
# using a recursive function.

#include <stdio.h>

// Recursive function to find GCD
int gcd(int a, int b) {
    if (b == 0) {
        return a;
    }
    return gcd(b, a % b);
}

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    if (num1 <= 0 || num2 <= 0) {
        printf("Please enter positive integers.\n");
    }
    else {
        int result = gcd(num1, num2);
        printf("The GCD of %d and %d is %d.\n", num1, num2, result);
    }

    return 0;
}
```