DAY 15-DAILY ASSIGNMENTS

26-11-2024

ANSU MARIUM SHIBU

1. Problem Statement:

Write a program that defines a custom data type Complex using typedef to represent a complex number with real and imaginary parts. Implement functions to:

Add two complex numbers.

Multiply two complex numbers.

Display a complex number in the format "a + bi".

Input Example

Enter first complex number (real and imaginary): 3 4

Enter second complex number (real and imaginary): 1 2

Output Example

Sum: 4 + 6i

Product: -5 + 10i

```c
#include<stdio.h>

typedef float complex;

int main(){
    complex real1,imag1;
    complex real2,imag2;
    complex sumreal,sumimag;
    complex productreal,productimag;

    printf("enter first real and imag num:");
    scanf("%f %f", &real1, &imag1);

    printf("enter second real and imag num:");
    scanf("%f %f", &real2, &imag2);

    sumreal=real1+real2;
    sumimag=imag1+imag2;

    productreal=real1*real2 - imag1*imag2;
    productimag=real1*imag2 + imag1*real2;

    printf("sum: %.2f + %.2fi\n", sumreal, sumimag);
    printf("product: %.2f + %.2fi\n", productreal, productimag);
}
```

```
su
PS D:\c progrms coding> gcc typedefassi1.c
PS D:\c progrms coding> ./a
enter first real and imag num:3 4
enter second real and imag num:1 2
sum: 4.00 + 6.00i
product: -5.00 + 10.00i
```

2. Typedef for Structures

Problem Statement:

Define a custom data type Rectangle using typedef to represent a rectangle with width and height as float values. Write functions to:

Compute the area of a rectangle.

Compute the perimeter of a rectangle.

Input Example:

Enter width and height of the rectangle: 5 10

Output Example:

Area: 50.00Perimeter: 30.00

typedefassi1.c typedefassi2.c structdynmicpoiassi2.c

typedefassi2.c > rectagle

```c
#include<stdio.h>

typedef struct Rectagle{
    float width;
    float height;
}rectangle;

int main(){
    rectangle rect;

    printf("enter the width:");
    scanf("%f",&rect.width);

    printf("enter heoght:");
    scanf("%f",&rect.height);

    float area=rect.width *rect.height;
    float perimeter=2*(rect.width +rect.height);

    printf("area:%.2f\n",area);
    printf("perimeter:%.2f\n",perimeter);
}
```

```
perimeter:30.00
PS D:\c progrms coding> gcc typedefassi2.c
PS D:\c progrms coding> ./a
enter the width:5
enter heoght:10
area:50.00
perimeter:30.00
PS D:\c progrms coding>
```

3. Simple Calculator Using Function Pointers

Problem Statement:

Write a C program to implement a simple calculator. Use function pointers to dynamically call functions for addition, subtraction, multiplication, and division based on user input.

Input Example:

Enter two numbers: 10 5

Choose operation (+, -, *, /): *

Output Example:

Result: 50

```c
#include<stdio.h>

void add(int,int);
void sub(int,int);
void mul(int,int);
void div(int,int);

int main(){
    void(*fun_ptr[])(int,int)={add,sub,mul,div};
    int a,b,choice;
    printf("enter fisrt num:");
    scanf("%d",&a);
    printf("enter sec num:");
    scanf("%d",&b);

    printf("choose op:\n");
    printf("1. Add\n2. Subtract\n3. Multiply\n4. Divide\n");
    printf("enter choice:");
    scanf("%d",&choice);

    if(choice>=1 && choice<=4){
        (*fun_ptr[choice-1])(a,b);
    }else{
        printf("invalis\n");
    }

}
void add(int a,int b){
    printf("sum:%d\n",a+b);
```

```c
int main(){
    printf("enter choice:");
    scanf("%d",&choice);

    if(choice>=1 && choice<=4){
        (*fun_ptr[choice-1])(a,b);
    }else{
        printf("invalis\n");
    }

}
void add(int a,int b){
    printf("sum:%d\n",a+b);
}
void sub(int a,int b){
    printf("sub:%d\n",a-b);
}
void mul(int a,int b){
    printf("mul:%d\n",a*b);
}
void div(int a,int b){
    if(b!=0){
        printf("div:%f\n",(float)a/b);
    }else{
        printf("invalid\n");
    }
}
```

```
1
PS D:\c progrms coding> gcc arrfunpoias1.c
PS D:\c progrms coding> ./a
enter fisrt num:3
enter sec num:4
choose op:
1. Add
2. Subtract
3. Multiply
4. Divide
enter choice:1
sum:7
PS D:\c progrms coding> gcc arrfunpoias1.c
PS D:\c progrms coding> ./a
enter fisrt num:20
enter sec num:5
choose op:
1. Add
2. Subtract
3. Multiply
4. Divide
enter choice:4
div:4.000000
PS D:\c progrms coding>
```

4. Array Operations Using Function Pointers

Problem Statement:

Write a C program that applies different operations to an array of integers using function pointers. Implement operations like finding the maximum, minimum, and sum of elements.

Input Example:

Enter size of array: 4

Enter elements: 10 20 30 40

Choose operation (1 for Max, 2 for Min, 3 for Sum): 3

Output Example:

Result: 100

funpoiass2.c > ...

```c
#include<stdio.h>

int find_max(int arr[], int size);
int find_min(int arr[], int size);
int find_sum(int arr[], int size);

int main() {
    int choice, size;

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

    int arr[size];

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

    printf("Choose operation (1 for Max, 2 for Min, 3 for Sum): ");
    scanf("%d", &choice);

    int (*fun_ptr[])(int[], int) = {find_max, find_min, find_sum};

    if (choice >= 1 && choice <= 3) {
        int result = (*fun_ptr[choice - 1])(arr, size);
        printf("Result: %d\n", result);
    } else {
        printf("Invalid choice\n");
```

```c
int find_max(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

int find_min(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    return min;
}

int find_sum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}
```

```
PS D:\c progrms coding> gcc arrfunpoiass2.c
PS D:\c progrms coding> ./a
Enter size: 3
Enter elements: 1 2 3
Choose operation (1 for Max, 2 for Min, 3 for Sum): 1
Result: 3
PS D:\c progrms coding> gcc arrfunpoiass2.c
PS D:\c progrms coding> ./a
Enter size: 3
Enter elements: 123
3 4
Choose operation (1 for Max, 2 for Min, 3 for Sum): 3
Result: 130
PS D:\c progrms coding>
```

5.

Event System Using Function Pointers

Problem Statement:

Write a C program to simulate a simple event system. Define three events: onStart, onProcess, and onEnd. Use function pointers to call appropriate event handlers dynamically based on user selection.

Input Example:

Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): 1

Output Example:

Event: onStart

Starting the process...

```c
#include<stdio.h>

void onstart();
void onprocess();
void onend();

int main(){
    void (*fun_ptr[3])()={onstart,onprocess,onend};

    int choice;
    printf("Choose an event:\n");
    printf("1. Start\n 2. Process\n 3. End\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);

    if(choice>=1 && choice<=3){
        fun_ptr[choice-1]();
    }else{
        printf("invalid");
    }

}
void onstart(){
    printf("Event: Start\n");
    printf("Starting the process...\n");
```

```
    }else{
        printf("invalid");
    }

}
void onstart(){
    printf("Event: Start\n");
    printf("Starting the process...\n");

}
void onprocess(){
    printf("Event: onprocess\n");
     printf("Processing the data...\n");

}
void onend(){
    printf("Event: end\n");
    printf("Ending the process...\n");

}
```

```
PS D:\c progrms coding> gcc arrfunpoias3.c
PS D:\c progrms coding> ./a
Choose an event:
1. Start
 2. Process
 3. End
Enter your choice: 1
Event: Start
Starting the process...
PS D:\c progrms coding>
```

6.Write a C program to perform matrix operations using function pointers. Implement functions to add, subtract, and multiply matrices. Pass the function pointer to a wrapper function to perform the desired operation.

Input Example:

Enter matrix size (rows and columns): 2 2

Enter first matrix:

1 2

3 4

Enter second matrix:

5 6

7 8

Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1

Output Example:

Result:

6 8

10 12

```c
#include <stdio.h>

void add_matrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]);
void sub_matrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]);
void mul_matrices(int rows1, int cols1, int matrix1[rows1][cols1], int rows2, int cols2, int matrix2[rows2][cols2], int result[rows1][cols

int main() {
    int rows1, cols1, rows2, cols2, choice;

    printf("Enter the number of rows for the first matrix: ");
    scanf("%d", &rows1);
    printf("Enter the number of columns for the first matrix: ");
    scanf("%d", &cols1);

    int matrix1[rows1][cols1];

    printf("Enter the elements of the first matrix:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    printf("Enter the number of rows for the second matrix: ");
```

```c
    int matrix2[rows2][cols2];

    printf("Enter the elements of the second matrix:\n");
    for (int i = 0; i < rows2; i++) {
        for (int j = 0; j < cols2; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }

    if (cols1 != rows2) {
        printf("Matrix multiplication not possible. Number of columns in the first matrix must equal the number of rows in the second ma
        return 0;
    }

    int result[rows1][cols2];

    printf("Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): ");
    scanf("%d", &choice);

    if (choice == 1 || choice == 2) {
        if (rows1 != rows2 || cols1 != cols2) {
            printf("Matrix addition or subtraction requires matrices to have the same dimensions.\n");
```

```c
    if (choice == 1 || choice == 2) {
        if (rows1 != rows2 || cols1 != cols2) {
            printf("Matrix addition or subtraction requires matrices to have the same dimensions.\n");
            return 0;
        }
    }

    switch (choice) {
        case 1:
            add_matrices(rows1, cols1, matrix1, matrix2, result);
            break;
        case 2:
            sub_matrices(rows1, cols1, matrix1, matrix2, result);
            break;
        case 3:
            mul_matrices(rows1, cols1, matrix1, rows2, cols2, matrix2, result);
            break;
        default:
            printf("Invalid choice.\n");
            return 0;
    }
```

```c
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < (choice == 3 ? cols2 : cols1); j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}

void add_matrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

void sub_matrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
```

```
PS D:\c progrms coding> gcc arfunpoi4.c
PS D:\c progrms coding> ./a
Enter the number of rows: 2 2
Enter the number of columns: Enter elements of the first matrix:
2 3
4 5
Enter elements of the second matrix:
2 4
5 6
Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1
Result:
4 7
236099722 531
PS D:\c progrms coding> gcc arfunpoi4.c
PS D:\c progrms coding> ./a
Enter the number of rows for the first matrix: 2
Enter the number of columns for the first matrix: 2
Enter the elements of the first matrix:
1 2
3 4
Enter the number of rows for the second matrix: 2
Enter the number of columns for the second matrix: 2
Enter the elements of the second matrix:
5 6
7 8
Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1
Result:
6 8
10 12
PS D:\c progrms coding>
```

7. Problem Statement: Vehicle Management System

Write a C program to manage information about various vehicles. The program should demonstrate the following:

Structures: Use structures to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year.

Unions: Use a union to represent type-specific attributes, such as:

Car: Number of doors and seating capacity.

Bike: Engine capacity and type (e.g., sports, cruiser).

Truck: Load capacity and number of axles.

Typedefs: Define meaningful aliases for complex data types using typedef (e.g., for the structure and union types).

Bitfields: Use bitfields to store flags for vehicle features like airbags, ABS, and sunroof.

Function Pointers: Use a function pointer to dynamically select a function to display specific information about a vehicle based on its type.

Requirements

Create a structure Vehicle that includes:

A char array for the manufacturer name.

An integer for the model year.

A union VehicleDetails for type-specific attributes.

A bitfield to store vehicle features (e.g., airbags, ABS, sunroof).

A function pointer to display type-specific details.

Write functions to:

Input vehicle data, including type-specific details and features.

Display all the details of a vehicle, including the type-specific attributes.

Set the function pointer based on the vehicle type.

Provide a menu-driven interface to:

Add a vehicle.

Display vehicle details.

Exit the program.

Example Input/Output

Input:

1. Add Vehicle

2. Display Vehicle Details

3. Exit

Enter your choice: 1

Enter vehicle type (1: Car, 2: Bike, 3: Truck): 1

Enter manufacturer name: Toyota

Enter model year: 2021

Enter number of doors: 4

Enter seating capacity: 5

Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): 1 1 0

1. Add Vehicle

2. Display Vehicle Details

3. Exit

Enter your choice: 2

Output:

Manufacturer: Toyota

Model Year: 2021

Type: Car

Number of Doors: 4

Seating Capacity: 5

Features: Airbags: Yes, ABS: Yes, Sunroof: No

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

typedef union {
    struct {
        int doors;
        int seat_capacity;
    } car;

    struct {
        int engine_capacity;
        char type[10];
    } bike;

    struct {
        int load_capacity;
        int axles;
    } truck;
} vd;

typedef struct {
    unsigned airbags : 1;
    unsigned ABS : 1;
    unsigned sunroof : 1;
} ft;
```

```
    unsigned ABS : 1;
    unsigned sunroof : 1;
} ft;

typedef struct {
    char manufacturer[50];
    int model_year;
    char type[10];
    vd details;
    ft features;
} veh;

void input_vehicle(veh *v);
void display_vehicle(veh v);

int main() {
    veh vehicles[10];
    int count = 0, choice;

    while (1) {
        printf("\n--- Vehicle Management System ---\n");
        printf("1. Add Vehicle\n2. Display Vehicle Details\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
    veh vehicles[10];
    int count = 0, choice;

    while (1) {
        printf("\n--- Vehicle Management System ---\n");
        printf("1. Add Vehicle\n2. Display Vehicle Details\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            if (count < 10) {
                input_vehicle(&vehicles[count]);
                count++;
            } else {
                printf("Vehicle list is full!\n");
            }
        } else if (choice == 2) {
            for (int i = 0; i < count; i++) {
                display_vehicle(vehicles[i]);
            }
        } else if (choice == 3) {
            break;
        } else {
            printf("Invalid choice. Please try again.\n");
```

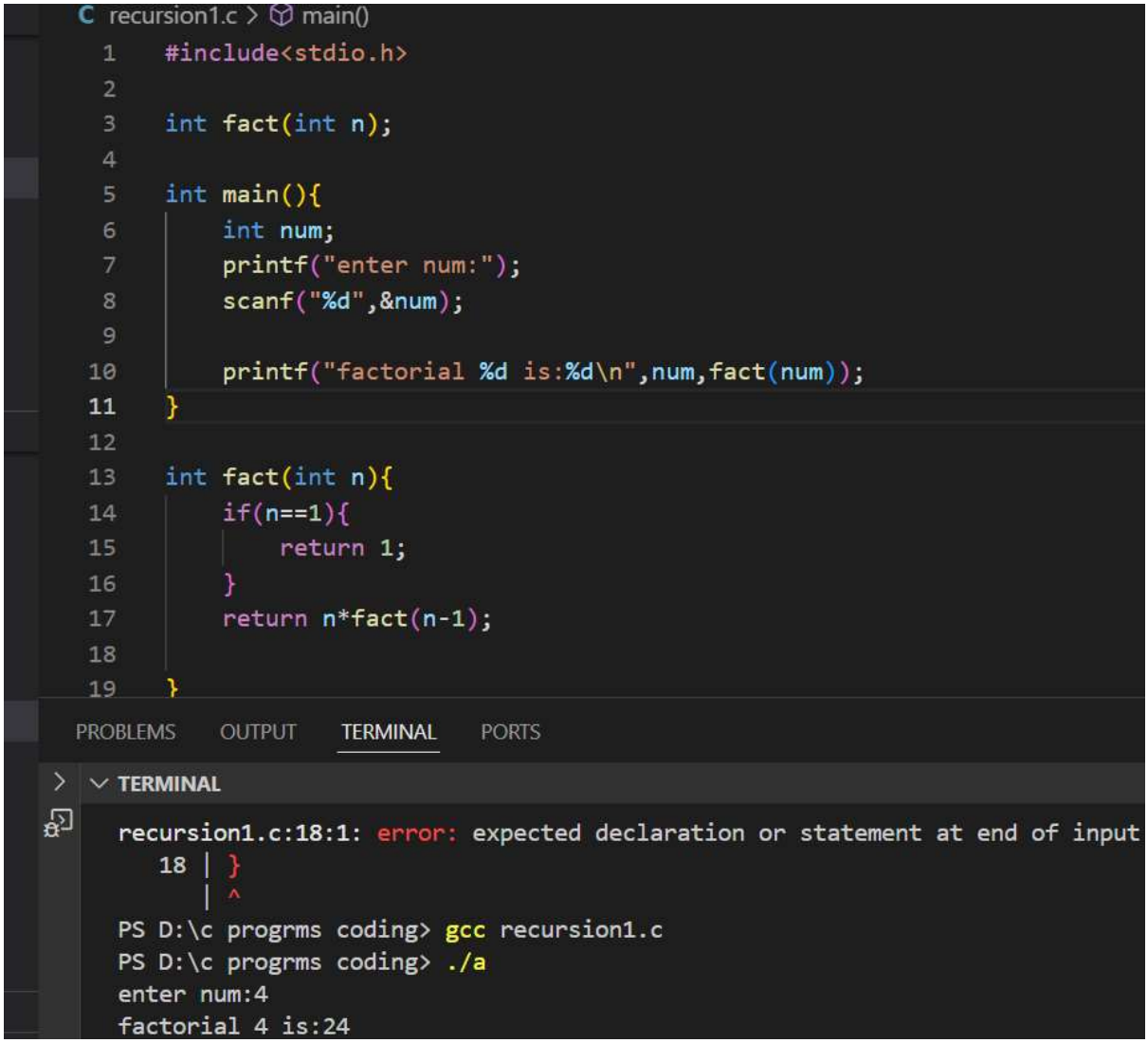
```
    }
    return 0;
}

void input_vehicle(veh *v) {
    printf("Enter manufacturer: ");
    scanf("%s", v->manufacturer);
    printf("Enter model year: ");
    scanf("%d", &v->model_year);

    printf("Enter vehicle type (car/bike/truck): ");
    scanf("%s", v->type);

    if (strcmp(v->type, "car") == 0) {
        printf("Enter number of doors: ");
        scanf("%d", &v->details.car.doors);
        printf("Enter seating capacity: ");
        scanf("%d", &v->details.car.seat_capacity);
    } else if (strcmp(v->type, "bike") == 0) {
        printf("Enter engine capacity (in cc): ");
        scanf("%d", &v->details.bike.engine_capacity);
        printf("Enter type (sports/cruiser): ");
        scanf("%s", v->details.bike.type);
```

```
68  void input_vehicle(veh *v) {
77      if (strcmp(v->type, "car") == 0) {
82      } else if (strcmp(v->type, "bike") == 0) {
83          printf("Enter engine capacity (in cc): ");
84          scanf("%d", &v->details.bike.engine_capacity);
85          printf("Enter type (sports/cruiser): ");
86          scanf("%s", v->details.bike.type);
87      } else if (strcmp(v->type, "truck") == 0) {
88          printf("Enter load capacity (in tons): ");
89          scanf("%d", &v->details.truck.load_capacity);
90          printf("Enter number of axles: ");
91          scanf("%d", &v->details.truck.axles);
92      }
93
94      unsigned temp_airbags, temp_ABS, temp_sunroof;
95
96      printf("Does the vehicle have airbags? (1 for Yes, 0 for No): ");
97      scanf("%u", &temp_airbags);
98      v->features.airbags = temp_airbags;
99
00      printf("Does the vehicle have ABS? (1 for Yes, 0 for No): ");
01      scanf("%u", &temp_ABS);
02      v->features.ABS = temp_ABS;
03
04      printf("Does the vehicle have a sunroof? (1 for Yes, 0 for No): ");
```

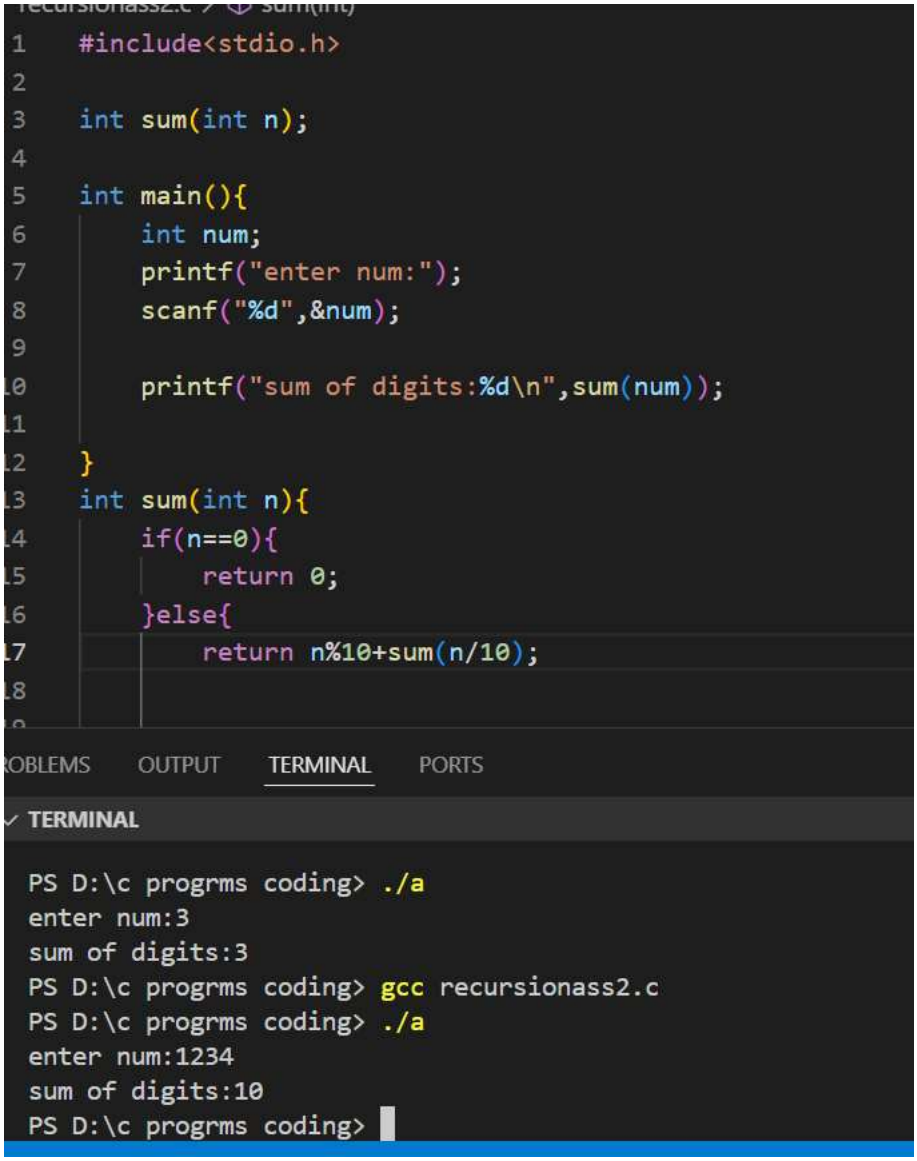
```c
void input_vehicle(veh *v) {

    printf("Does the vehicle have ABS? (1 for Yes, 0 for No): ");
    scanf("%u", &temp_ABS);
    v->features.ABS = temp_ABS;

    printf("Does the vehicle have a sunroof? (1 for Yes, 0 for No): ");
    scanf("%u", &temp_sunroof);
    v->features.sunroof = temp_sunroof;
}

void display_vehicle(veh v) {
    printf("\n--- Vehicle Details ---\n");
    printf("Manufacturer: %s\n", v.manufacturer);
    printf("Model Year: %d\n", v.model_year);
    printf("Type: %s\n", v.type);

    if (strcmp(v.type, "car") == 0) {
        printf("Number of doors: %d\n", v.details.car.doors);
        printf("Seating capacity: %d\n", v.details.car.seat_capacity);
    } else if (strcmp(v.type, "bike") == 0) {
        printf("Engine capacity: %d cc\n", v.details.bike.engine_capacity);
        printf("Type: %s\n", v.details.bike.type);
    } else if (strcmp(v.type, "truck") == 0) {
        printf("Load capacity: %d tons\n", v.details.truck.load_capacity);
        printf("Number of axles: %d\n", v.details.truck.axles);
    }

    printf("Features: Airbags: %u, ABS: %u, Sunroof: %u\n",
           v.features.airbags, v.features.ABS, v.features.sunroof);
}
```

```
--- Vehicle Management System ---
1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 1
Enter manufacturer: toyota
Enter model year: 2021
Enter vehicle type (car/bike/truck): car
Enter number of doors: 4
Enter seating capacity: 5
Does the vehicle have airbags? (1 for Yes, 0 for No): 1
Does the vehicle have ABS? (1 for Yes, 0 for No): 1
Does the vehicle have a sunroof? (1 for Yes, 0 for No): 1

--- Vehicle Management System ---
1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 2

--- Vehicle Details ---
Manufacturer: toyota
Model Year: 2021
Type: car
Number of doors: 4
Seating capacity: 5
Features: Airbags: 1, ABS: 1, Sunroof: 1
```

8. WAP to find out the factorial of a number using recursion



9. WAP to find the sum of digits of a number using recursion.



```c
#include<stdio.h>

int sum(int n);

int main(){
    int num;
    printf("enter num:");
    scanf("%d",&num);

    printf("sum of digits:%d\n",sum(num));

}
int sum(int n){
    if(n==0){
        return 0;
    }else{
        return n%10+sum(n/10);
```

```
OBLEMS  OUTPUT  TERMINAL  PORTS
TERMINAL
PS D:\c progrms coding> ./a
enter num:3
sum of digits:3
PS D:\c progrms coding> gcc recursionass2.c
PS D:\c progrms coding> ./a
enter num:1234
sum of digits:10
PS D:\c progrms coding>
```

10. With Recursion Findout the maximum number in a given array

```c
#include<stdio.h>

int max(int arr[],int n);

int main(){
    int n;
    printf("enetr size:");
    scanf("%d",&n);
    int arr[n];
    printf("enter num:");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);

    }
    int maxi=max(arr,n);
    printf("max :%d",maxi);
}
int max(int arr[],int n){
    if(n==1){
        return arr[0];
    }
    int max_res=max(arr+1,n-1);
    if(arr[0]>max_res){
        return arr[0];
    }else{
        return max_res;
}

    }
```

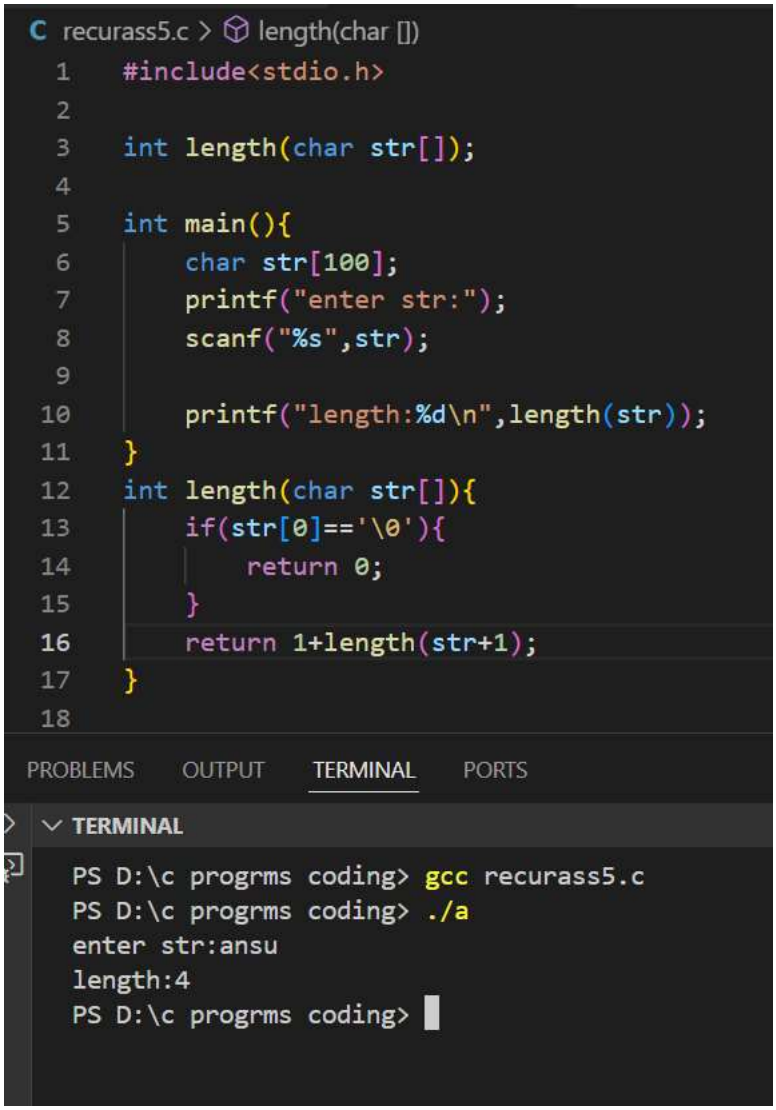
```
PS D:\c progrms coding> gcc recurssionass3.c
PS D:\c progrms coding> ./a
enetr size:5
enter num:3
4
1
7
2
max :7
PS D:\c progrms coding>
```

11. With recurion calculate the power of a given number

```c
int power(int base,int expo);

int main(){
    int base,expo;
    printf("enter base:");
    scanf("%d",&base);
    printf("entr expo:");
    scanf("%d",&expo);

     printf("%d raised to the power of %d is: %d\n", base, expo, power(base, expo));


}

int power(int base,int expo){
    if(expo==0){
        return 1;
    }
    return base*power(base,expo-1);
}
```

```
PS D:\c progrms coding> gcc recurssionass4.c
PS D:\c progrms coding> ./a
enter base:2
entr expo:3
2 raised to the power of 3 is: 8
PS D:\c progrms coding>
```

12. With Recursion calculate the length of a string.



13. With recursion revrsal of a string

```c
#include<stdio.h>

void rev(char str[],int index);

int main(){
    char str[100];
    printf("enter str:");
    scanf("%s",str);

    printf("Reversed string: ");
    rev(str, 0);
    printf("\n");
}
void rev(char str[],int index){
    if (str[index] == '\0') {
        return;
    }
    rev(str, index + 1);
    printf("%c", str[index]);
}
```

```
LEMS   OUTPUT   TERMINAL   PORTS
TERMINAL
xewin.c:67:(.text.startup+0xc5): undefined reference to `WinMain'
collect2.exe: error: ld returned 1 exit status
PS D:\c progrms coding> gcc recurass6.c
PS D:\c progrms coding> ./a
enter str:ansu
Reversed string: usna
PS D:\c progrms coding>
```