DAY15_ASSIGNMENT_ABHIRAMI

/*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
***/**

```
#include<stdio.h>
typedef struct {
    int real;
    int imaginary;
}Complex;

Complex addComplex(Complex c1, Complex c2);
Complex multiplyComplex(Complex c1, Complex c2);
void displayComplex(Complex c);
int main() {
    Complex c1, c2, sum, product;

       printf("Enter first complex number (real and imaginary): ");
    scanf("%d %d", &c1.real, &c1.imaginary);

      printf("Enter second complex number (real and imaginary): ");
    scanf("%d %d", &c2.real, &c2.imaginary);

       sum = addComplex(c1, c2);
    product = multiplyComplex(c1, c2);

        printf("Sum: ");
    displayComplex(sum);
    printf("Product: ");
    displayComplex(product);

    return 0;
}
```

```
Complex addComplex(Complex c1, Complex c2) {
    Complex result;
    result.real = c1.real + c2.real;
    result.imaginary = c1.imaginary + c2.imaginary;
    return result;
}

Complex multiplyComplex(Complex c1, Complex c2) {
    Complex result;
    result.real = c1.real * c2.real - c1.imaginary * c2.imaginary;
    result.imaginary = c1.real * c2.imaginary + c1.imaginary * c2.real;
    return result;
}

void displayComplex(Complex c) {
    if (c.imaginary >= 0)
        printf("%d + %di\n", c.real, c.imaginary);
    else
        printf("%d - %di\n", c.real, -c.imaginary);
}
```

/*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: 50.00
Perimeter: 30.00*/

```
#include <stdio.h>
typedef struct {
    float width;
    float height;
} Rectangle;

float computeArea(Rectangle rect);
float computePerimeter(Rectangle rect);

int main() {
    Rectangle rect;
```

```
    printf("Enter width and height of the rectangle: ");
    scanf("%f %f", &rect.width, &rect.height);


    float area = computeArea(rect);
    float perimeter = computePerimeter(rect);


    printf("Area: %.2f\n", area);
    printf("Perimeter: %.2f\n", perimeter);

    return 0;
}
float computeArea(Rectangle rect) {
    return rect.width * rect.height;
}


float computePerimeter(Rectangle rect) {
    return 2 * (rect.width + rect.height);
}
```

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>

float add(float a, float b);
float subtract(float a, float b);
float multiply(float a, float b);
float divide(float a, float b);
int main() {
    float num1, num2, result;
    char operation;
    float (*operationFunc)(float, float);



    printf("Enter two numbers: ");
    scanf("%f %f", &num1, &num2);
    printf("Choose operation (+, -, *, /): ");
    scanf(" %c", &operation);

    // Assign appropriate function to function pointer
    switch (operation) {
        case '+': operationFunc = add; break;
        case '-': operationFunc = subtract; break;
        case '*': operationFunc = multiply; break;
        case '/': operationFunc = divide; break;
        default:
```

```
            printf("Invalid operation.\n");
            return 1;
    }

    // Call the function dynamically
    result = operationFunc(num1, num2);
    printf("Result: %.2f\n", result);

    return 0;
}

// Function declarations
float add(float a, float b) {
    return a + b;
    }
float subtract(float a, float b) {
    return a - b;

}
float multiply(float a, float b) {
    return a * b;

}
float divide(float a, float b) {
```

```
    return b != 0 ? a / b : 0;


}
```

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

```
#include <stdio.h>

int findMax(int arr[], int size);

int findMin(int arr[], int size);

int findSum(int arr[], int size);


int main() {

    int size, choice, result;

    printf("Enter size of array: ");

    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);

// Define function pointer
int (*operationFunc)(int[], int);

// Assign appropriate function
switch (choice) {
    case 1: operationFunc = findMax; break;
    case 2: operationFunc = findMin; break;
    case 3: operationFunc = findSum; break;
    default:
        printf("Invalid choice.\n");
        return 1;
}

result = operationFunc(arr, size);
printf("Result: %d\n", result);
```

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

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...

```
#include <stdio.h>
void onStart();
void onProcess();
void onEnd();
int main() {
    int choice;
    printf("Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): ");
    scanf("%d", &choice);

    // Define function pointer
    void (*eventHandler)();

    // Assign appropriate handler
    switch (choice) {
        case 1: eventHandler = onStart;
        break;
        case 2: eventHandler = onProcess;
        break;
        case 3: eventHandler = onEnd;
        break;
        default:
            printf("Invalid event.\n");
            return 1;
    }


    eventHandler();
```

```
    return 0;
}
void onStart() {
    printf("Event: onStart\nStarting the process...\n");
    }
void onProcess() {
    printf("Event: onProcess\nProcessing...\n");

}
void onEnd() {
    printf("Event: onEnd\nEnding the process...\n");

}
```

Matrix Operations with Function Pointers

Problem Statement:
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
```

```
#include <stdio.h>

void addMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]);
void subtractMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]);
```

```
void multiplyMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]);


void displayMatrix(int rows, int cols, int matrix[rows][cols]);


int main() {
    int rows, cols, choice;
    printf("Enter matrix size (rows and columns): ");
    scanf("%d %d", &rows, &cols);

    int a[rows][cols], b[rows][cols], result[rows][cols];

    printf("Enter first matrix:\n");
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            scanf("%d", &a[i][j]);

    printf("Enter second matrix:\n");
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            scanf("%d", &b[i][j]);

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

    // Define function pointer
    void (*matrixOperation)(int, int, int[rows][cols], int[rows][cols], int[rows][cols]);

    // Assign appropriate function
    switch (choice) {
        case 1: matrixOperation = addMatrices; break;
        case 2: matrixOperation = subtractMatrices; break;
        case 3: matrixOperation = multiplyMatrices; break;
        default:
            printf("Invalid choice.\n");
            return 1;
    }


    matrixOperation(rows, cols, a, b, result);
```

```
    printf("Result:\n");
    displayMatrix(rows, cols, result);

    return 0;
}

void addMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]) {
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            result[i][j] = a[i][j] + b[i][j];
}

void subtractMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]) {
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            result[i][j] = a[i][j] - b[i][j];
}

void multiplyMatrices(int rows, int cols, int a[rows][cols], int b[rows][cols], int result[rows][cols]) {
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++) {
            result[i][j] = 0;
            for (int k = 0; k < cols; k++)
                result[i][j] += a[i][k] * b[k][j];
        }
}

void displayMatrix(int rows, int cols, int matrix[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++)
            printf("%d ", matrix[i][j]);
        printf("\n");
    }
}
```

Problem Statement: Vehicle Management System

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

1. **Structures**: Use structures to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year.
2. **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.
3. **Typedefs**: Define meaningful aliases for complex data types using typedef (e.g., for the structure and union types).
4. **Bitfields**: Use bitfields to store flags for vehicle features like **airbags**, **ABS**, and **sunroof**.
5. **Function Pointers**: Use a function pointer to dynamically select a function to display specific information about a vehicle based on its type.

Requirements

1. 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.
2. 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.
3. 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

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

//enum for vehicle type
typedef enum { CAR = 1, BIKE, TRUCK } vehicle_type;

//structure to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year
typedef struct {
    vehicle_type type;
    char manufacturer_name[50];
    int model_year;
} vehicle;

//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
typedef union {
    struct {
        int number_of_doors;
        int seating_capacity;
    } car;
    struct {
        int engine_capacity;
        char bike_type[50];
```

```
    } bike;
    struct {
        int load_capacity;
        int number_of_axles;
    } truck;
} vehicle_details;

//bitfield to store vehicle features (e.g., airbags, ABS, sunroof)
typedef struct {
    unsigned int airbags:1;
    unsigned int abs:1;
    unsigned int sunroof:1;
} vehicle_features;

void add_vehicle(vehicle *, vehicle_details *, vehicle_features *);
void display_vehicle(vehicle *,vehicle_details *,vehicle_features *);

int main(){
    vehicle v;
    vehicle_details vd;
    vehicle_features vf;

    void (*vehicle_ptr[])(vehicle *, vehicle_details *, vehicle_features *) ={add_vehicle,
display_vehicle};
    int choice;

    while(1) {
        printf("\n1. Add Vehicle\n");
        printf("2. Display Vehicle Details\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        printf("\n");

        if (choice == 3) {
            printf("Exiting...\n");
            break;
        }

        if (choice >= 1 && choice <= 2) {
            vehicle_ptr[choice - 1](&v, &vd, &vf);
        } else {
            printf("Invalid choice. Please try again.\n");
        }
```

```
    }
    return 0;
}

void add_vehicle(vehicle *v, vehicle_details *vd, vehicle_features *vf) {
    int type;
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &type);
    if (type < 1 || type > 3) {
    printf("Invalid vehicle type. Please try again.\n");
    return;
    }
    v->type = type;
    printf("Enter manufacturer name: ");
    scanf("%s", v->manufacturer_name);
    printf("Enter model year: ");
    scanf("%d", &v->model_year);

    switch (v->type) {
        case CAR:
            printf("Enter number of doors: ");
            scanf("%d", &vd->car.number_of_doors);
            printf("Enter seating capacity: ");
            scanf("%d", &vd->car.seating_capacity);
            break;
        case BIKE:
            printf("Enter engine capacity: ");
            scanf("%d", &vd->bike.engine_capacity);
            printf("Enter bike type: ");
            scanf("%s", vd->bike.bike_type);
            break;
        case TRUCK:
            printf("Enter load capacity: ");
            scanf("%d", &vd->truck.load_capacity);
            printf("Enter number of axles: ");
            scanf("%d", &vd->truck.number_of_axles);
            break;
    }

    int airbags, abs, sunroof;
    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%u %u %u", &airbags, &abs, &sunroof);
    vf->airbags = airbags ? 1 : 0;
    vf->abs = abs ? 1 : 0;
```

```
    vf->sunroof = sunroof ? 1 : 0;
}

void display_vehicle(vehicle *v, vehicle_details *vd, vehicle_features *vf) {
    printf("Manufacturer: %s\n", v->manufacturer_name);
    printf("Model Year: %d\n", v->model_year);

    switch (v->type) {
        case CAR:
            printf("Type: Car\n");
            printf("Number of Doors: %d\n", vd->car.number_of_doors);
            printf("Seating Capacity: %d\n", vd->car.seating_capacity);
            break;
        case BIKE:
            printf("Type: Bike\n");
            printf("Engine Capacity: %d\n", vd->bike.engine_capacity);
            printf("Bike Type: %s\n", vd->bike.bike_type);
            break;
        case TRUCK:
            printf("Type: Truck\n");
            printf("Load Capacity: %d\n", vd->truck.load_capacity);
            printf("Number of Axles: %d\n", vd->truck.number_of_axles);
            break;
        default:
            printf("No details to display.\n");
            break;
    }
    printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n", vf->airbags? "Yes" : "No", vf->abs?
"Yes" : "No", vf->sunroof? "Yes" : "No");
    printf("\n");
}
```

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

```
#include <stdio.h>



int factorial(int n) {

    if (n == 0 || n == 1) {

        return 1;     }
```

```
    return n * factorial(n - 1);
}

int main() {
    int num;
    printf("Enter a number to find its factorial: ");
    scanf("%d", &num);
    printf("Factorial of %d is %d\n", num, factorial(num));
    return 0;
}
```

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

```
#include <stdio.h>

int sumOfDigits(int n) {
    if (n == 0) {
        return 0; // Base case
    }
    return (n % 10) + sumOfDigits(n / 10);
}

int main() {
```

```
    int num;
    printf("Enter a number to find the sum of its digits: ");
    scanf("%d", &num);
    printf("Sum of digits of %d is %d\n", num, sumOfDigits(num));
    return 0;
}
```

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

ower of a given number

```
#include <stdio.h>

int findMax(int arr[], int n) {
    if (n == 1) {
        return arr[0];    }
    int maxRest = findMax(arr, n - 1);
    return (arr[n - 1] > maxRest) ? arr[n - 1] : maxRest;
}

int main() {
    int arr[] = {3, 5, 7, 2, 8, 6};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Maximum number in the array is %d\n", findMax(arr, n));
    return 0;
```

```
}
```

4. With recurion calculate the power of a given number

```
 #include <stdio.h>

// Recursive function to calculate power
int power(int base, int exp) {
    if (exp == 0) {
        return 1;
    }
    return base * power(base, exp - 1);
}

int main() {
    int base, exp;
    printf("Enter base and exponent: ");
    scanf("%d %d", &base, &exp);
    printf("%d^%d is %d\n", base, exp, power(base, exp));
    return 0;
}
```

5. With Recursion calculate the length of a string.

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

```
int stringLength(char str[]) {
    if (*str == '\0') {
        return 0;    }
    return 1 + stringLength(str + 1);
}

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("Length of the string is %d\n", stringLength(str));
    return 0;
}
```

6. With recursion revrsal of a string

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

void reverseString(char str[], int start, int end) {
    if (start >= end) {
        return;
    }
    char temp = str[start];
    str[start] = str[end];
    str[end] = temp;
    reverseString(str, start + 1, end - 1);
}
```

```
int main() {
    char str[100];
    printf("Enter a string to reverse: ");
    scanf("%s", str);
    int len = strlen(str);
    reverseString(str, 0, len - 1);
    printf("Reversed string is: %s\n", str);
    return 0;
}
```