ASSIGNMENT_DAY_15

```
// 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 {
    float real;
    float img;
} IntNumber;

IntNumber intAdd(IntNumber c1, IntNumber c2);
IntNumber intMultiply(IntNumber c1, IntNumber c2);
void displayInt(IntNumber c);
int main() {
    IntNumber c1, c2, sum, product;
    printf("Enter the real and imaginary parts of the first number: ");
    scanf("%f %f", &c1.real, &c1.img);
    printf("Enter the real and imaginary parts of the second number: ");
    scanf("%f %f", &c2.real, &c2.img);
    sum = intAdd(c1, c2);
    product = intMultiply(c1, c2);
    printf("Sum: ");
    displayInt(sum);

    printf("Product: ");
    displayInt(product);

    return 0;
}

IntNumber intAdd(IntNumber c1, IntNumber c2) {
    IntNumber result;
    result.real = c1.real + c2.real;
    result.img = c1.img + c2.img;
    return result;
}

IntNumber intMultiply(IntNumber c1, IntNumber c2) {
    IntNumber result;
    result.real = c1.real * c2.real - c1.img * c2.img;
    result.img = c1.real * c2.img + c1.img * c2.real;
    return result;
}

void displayInt(IntNumber c) {
    printf("%.2f + %.2fi\n", c.real, c.img);
}
```

```
#include <stdio.h>

// Define a custom data type for Rectangle
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);
    printf("Area: %.2f\n", computeArea(rect));
    printf("Perimeter: %.2f\n", computePerimeter(rect));

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

```c
    return 2 * (rect.width + rect.height);
}
```

```c
// 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

#include<stdio.h>


float add(float a, float b) ;
float sub(float a, float b) ;
float mul(float a, float b) ;
float div(float a, float b) ;

int main(){
    float(*operation)(float,float);
    float num1,num2;
    char operator;

    printf("enter the num 1");
    scanf("%f",&num1);
    printf("enter the num2");
    scanf("%f",&num2);
    printf("enter the operator");
    scanf(" %c",&operator);
    switch (operator)
    {
    case '+':
        operation =add;
```

```
            break;
        case '-':
            operation =sub;
            break;
        case '*':
            operation =mul;
            break;
        case '/':
            operation =div;
            break;



        default:
            break;
        }
        float result = operation(num1,num2);
        printf("Result: %.2lf\n", result);

    }
    float add(float a, float b) {
        return a + b;
    }

    float sub(float a, float b) {
        return a - b;
    }

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

    float div(float a, float b) {
        if (b == 0) {
            printf("Error: Division by zero!\n");
            return 0;
        }
        return a / b;
    }
```

```
// 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 (*operations[])(int[], int) = {findMax, findMin, findSum};
    int size, choice;
    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);
    int result;
    switch (choice) {
        case 1:
            result = operations[0](arr, size);
            break;
        case 2:
            result = operations[1](arr, size);
            break;
        case 3:
            result = operations[2](arr, size);
            break;
        default:
            printf("Invalid choice!\n");
            return 1;
    }
    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(){
    void(*operator)(void);
    int n;
    printf("enter the choice");
    scanf("%d",&n);
    switch (n)
    {
    case 1:
        operator =onstart;


        break;
    case 2:
        operator = onprocess;


        break;
    case 3:
        operator =onend;


        break;

    default:
        break;
    }
    operator();
}
void onstart(){
    printf("on start");
}
void onprocess(){
    printf("on process");
}
void onend(){
    printf("on end");
}
```

```
#include <stdio.h>

// Function declarations
void addMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);
void subtractMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);
void multiplyMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);

int main() {
    int rows, cols, choice;
    int a[10][10], b[10][10], result[10][10];
    void (*operation)(int[][10], int[][10], int[][10], int, int);

    // Input matrix dimensions
    printf("Enter matrix size (rows and columns): ");
    scanf("%d %d", &rows, &cols);

    // Input matrices
    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]);
        }
    }

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

    // Perform the selected operation using a function pointer

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

    // Call the selected function
    operation(a, b, result, rows, cols);


    printf("Result:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
```

```
}

// Function to add matrices
void addMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = a[i][j] + b[i][j];
        }
    }
}

// Function to subtract matrices
void subtractMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = a[i][j] - b[i][j];
        }
    }
}

// Function to multiply matrices
void multiplyMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = 0; // Initialize the result element to 0
            for (int k = 0; k < cols; k++) {
                result[i][j] += a[i][k] * b[k][j]; // Multiply and accumulate the sum
            }
        }
    }
}
```

```
// WAP to calculate the sum of first n natural numbers using recursion

#include<stdio.h>
int sumNatural(int n);
int main(){
    int n;
    printf("enter the limit till which the summation of natural");
    scanf("%d",&n);
    printf("\n");

    int sum =sumNatural(n);
    printf("sum=%d",sum);
    return 0;
}

int sumNatural(int n){
    int res =0;
    if(n==0){
        return 0;
    }
    // recursive call

    res = n+sumNatural(n-1);
    // printf("%d",res);
}



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

#include<stdio.h>

int sumof(int n);

int main(){
int n;
printf("enter the number");
scanf("%d",&n);
int result =sumof(n);
printf("the result is =%d",result);

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

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

```
#include<stdio.h>

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


int main(){
int n;
printf("enter the number");
scanf("%d",&n);
int arr[n];
printf("enter the elements of the array");
for(int i=0;i<n;i++){
    scanf("%d",&arr[i]);
}
int max = findMax(arr, n);
printf("the result is =%d",max);

}
int findMax(int arr[],int n){
    int res =0;
    if(n==1){
        return arr[0];
    }
     int maxInRest = findMax(arr, n - 1);
        if (arr[n - 1] > maxInRest) {
        return arr[n - 1];
    } else {
        return maxInRest;
    }


}
```

```
//With recurion calculate the power of a given number
#include<stdio.h>

int power(int base,int exponent);

int main(){

    int base,exponent;
    printf("enter the base");
    scanf("%d",&base);
    printf("enter the exponent");
    scanf("%d",&exponent);
    int result=power(base,exponent);
    printf("the result is %d",result);
}
int power(int base,int exponent){
    int res=0;
    if(exponent==0){
        return 1;
    }
    res =base*power(base,exponent-1);

}
```

```
//5. With Recursion calculate the length of a string.

#include<stdio.h>
int strlength(char arr[]);

int main(){
    char arr[20];
    printf("enter the string");
    scanf("%[^\n]",arr);
    int result=strlength(arr);
    printf("result =%d",result);

}
int strlength(char arr[]){
    int res=0;
    if(arr[0]=='\0'){
        return 0;
    }
    res=1+strlength(arr+1);

}
```

```
#include <stdio.h>

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

int main() {
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);

    printf("Reversed string: ");
    reverseString(str, 0);
    printf("\n");

    return 0;
}

void reverseString(char str[], int index) {
    // Base case: if we reach the end of the string, return
    if (str[index] == '\0') {
        return;
    }
    // Recursive call
    reverseString(str, index + 1);
    // Print the character after the recursive call
    printf("%c", str[index]);
}
```

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.

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

typedef struct {
    unsigned int airbags : 1;
    unsigned int abs : 1;
    unsigned int sunroof : 1;
} Features;

typedef union {
    struct {
        int doors;
        int seating_capacity;
    } car;
    struct {
        int engine_capacity;
        char type[20];
    } bike;
    struct {
        int load_capacity;
        int axles;
    } truck;
} VehicleDetails;

typedef struct {
    char manufacturer[50];
    int model_year;
    int type;
    VehicleDetails details;
    Features features;
} Vehicle;

void addVehicle(Vehicle *vehicles, int *count);
void displayVehicles(Vehicle *vehicles, int *count);
```

```
int main() {
    Vehicle vehicles[10];
    int count = 0;
    int choice;

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

        switch (choice)
        {
        case 1:
            addVehicle(vehicles,&count);
            break;
        case 2:
           displayVehicles(vehicles,&count);
           break;
           case 3:
           exit(0);
           break;

        default:

        printf("wrong choice is entered");

            break;
        }
    }

    return 0;
}
void addVehicle(Vehicle vehicles[], int *count) {
    Vehicle *v = &vehicles[*count];
    printf("\nEnter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &v->type);
    printf("Enter manufacturer name: ");
    scanf("%s", v->manufacturer);
    printf("Enter model year: ");
    scanf("%d", &v->model_year);
    if (v->type == 1) {
        printf("Enter number of doors: ");
        scanf("%d", &v->details.car.doors);
        printf("Enter seating capacity: ");
        scanf("%d", &v->details.car.seating_capacity);
    } else if (v->type == 2) {
        printf("Enter engine capacity : ");
        scanf("%d", &v->details.bike.engine_capacity);
        printf("Enter type : ");
        scanf("%s", v->details.bike.type);
    } else if (v->type == 3) {
        printf("Enter load capacity : ");
        scanf("%d", &v->details.truck.load_capacity);
        printf("Enter number of axles: ");
        scanf("%d", &v->details.truck.axles);
    } else {
        printf("Invalid vehicle type. Skipping details entry.\n");
        return;
    }
    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    unsigned int tempAirbags, tempAbs, tempSunroof;
    scanf("%u %u %u", &tempAirbags, &tempAbs, &tempSunroof);
    v->features.airbags = tempAirbags;
    v->features.abs = tempAbs;
    v->features.sunroof = tempSunroof;

    (*count)++;
    printf("Vehicle added successfully!\n");
}

void displayVehicles(Vehicle vehicles[], int *count) {
    for (int i = 0; i < *count; i++) {
        Vehicle *v = &vehicles[i];
        printf("Vehicle %d:\n", i + 1);
        printf("Manufacturer: %s\n", v->manufacturer);
        printf("Model Year: %d\n", v->model_year);
        if (v->type == 1) {
            printf("Type: Car\n");
            printf("Number of Doors: %d\n", v->details.car.doors);
            printf("Seating Capacity: %d\n", v->details.car.seating_capacity);
        } else if (v->type == 2) {
            printf("Type: Bike\n");
            printf("Engine Capacity: %d \n", v->details.bike.engine_capacity);
            printf("Type: %s\n", v->details.bike.type);
        } else if (v->type == 3) {
            printf("Type: Truck\n");
            printf("Load Capacity: %d \n", v->details.truck.load_capacity);
            printf("Number of Axles: %d\n", v->details.truck.axles);
        } else {
            printf("Type: Unknown\n");
        }

        printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n",v->features.airbags ? "Yes" :
"No",v->features.abs ? "Yes" : "No",v->features.sunroof ? "Yes" : "No");
        // printf("%d",count);
    }
}
```