

1. Write a c program to add,subtract,multiply and divided two integers using a user defined type function with a return type.

Code:-

```
#include <stdio.h>

// Function prototypes
int add(int a, int b);
int subtract(int a, int b);
int multiply(int a, int b);
float divide(int a, int b);

int main() {
    int num1, num2;

    // Get user input
    printf("Enter the first integer: ");
    scanf("%d", &num1);
    printf("Enter the second integer: ");
    scanf("%d", &num2);

    // Perform operations and display results
    printf("Sum: %d\n", add(num1, num2));
    printf("Difference: %d\n", subtract(num1, num2));
    printf("Product: %d\n", multiply(num1, num2));

    if (num2 != 0) {
        printf("Quotient: %.2f\n", divide(num1, num2));
    } else {
        printf("Error: Division by zero is not allowed.\n");
    }

    return 0;
}

// Function definitions
int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}
```

```
int multiply(int a, int b) {  
    return a * b;  
}
```

```
float divide(int a, int b) {  
    return (float)a / b;  
}
```

Output:-

```
Enter the first integer: 5  
Enter the second integer: 8  
Sum: 13  
Difference: -3  
Product: 40  
Quotient: 0.63  
  
-----  
Process exited after 9.053 seconds with return value 0  
Press any key to continue . . .
```

2. Write a c program to calculate the sum of the first 20 natural numbers using a recursive function.

Code:-

```
#include <stdio.h>

// Recursive function to calculate the sum of the first n natural numbers
int sum_of_natural_numbers(int n) {
    // Base case: if n is 1, return 1
    if (n == 1) {
        return 1;
    }
    // Recursive case: n + sum of the first (n-1) natural numbers
    return n + sum_of_natural_numbers(n - 1);
}

int main() {
    int n = 20; // We want the sum of the first 20 natural numbers

    // Calculate the sum using the recursive function
    int sum = sum_of_natural_numbers(n);

    // Print the result
    printf("The sum of the first %d natural numbers is %d.\n", n, sum);

    return 0;
}
```

Output:-

```
The sum of the first 20 natural numbers is 210.
-----
Process exited after 0.05301 seconds with return value 0
Press any key to continue . . .
```

3. Write a c program to generate a Fibonacci series using a recursive function.

Code:-

```
#include <stdio.h>

// Recursive function to calculate the nth Fibonacci number
int fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main() {
    int num_terms;

    // Get the number of terms from the user
    printf("Enter the number of terms for the Fibonacci series: ");
    scanf("%d", &num_terms);

    // Print the Fibonacci series
    printf("Fibonacci series up to %d terms:\n", num_terms);
    for (int i = 0; i < num_terms; i++) {
        printf("%d ", fibonacci(i));
    }
    printf("\n");

    return 0;
}
```

Output:-

```
Enter the number of terms for the Fibonacci series: 4
Fibonacci series up to 4 terms:
0 1 1 2

-----
Process exited after 12.56 seconds with return value 0
Press any key to continue . . .
```

4. Write a c program to swap two integers using call-by-value and call-by-reference methods of passing arguments to a function.

Code:-

```
#include <stdio.h>

// Function to swap two integers using call-by-value
void swap_call_by_value(int a, int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
    printf("Inside swap_call_by_value: a = %d, b = %d\n", a, b);
}

// Function to swap two integers using call-by-reference
void swap_call_by_reference(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
    printf("Inside swap_call_by_reference: a = %d, b = %d\n", *a, *b);
}

int main() {
    int x, y;

    // Input values for x and y
    printf("Enter the first integer: ");
    scanf("%d", &x);
    printf("Enter the second integer: ");
    scanf("%d", &y);

    // Swap using call-by-value
    printf("Before swap_call_by_value: x = %d, y = %d\n", x, y);
    swap_call_by_value(x, y);
    printf("After swap_call_by_value: x = %d, y = %d\n", x, y);

    // Swap using call-by-reference
    printf("Before swap_call_by_reference: x = %d, y = %d\n", x, y);
    swap_call_by_reference(&x, &y);
    printf("After swap_call_by_reference: x = %d, y = %d\n", x, y);

    return 0;
}
```

Output:-

```
Enter the first integer: 6
Enter the second integer: 8
Before swap_call_by_value: x = 6, y = 8
Inside swap_call_by_value: a = 8, b = 6
After swap_call_by_value: x = 6, y = 8
Before swap_call_by_reference: x = 6, y = 8
Inside swap_call_by_reference: a = 8, b = 6
After swap_call_by_reference: x = 8, y = 6

-----
Process exited after 5.764 seconds with return value 0
Press any key to continue . . .
```

5. Write a c program to find the sum of the digits of the numbers using recursive function.

Code:-

```
#include <stdio.h>

// Recursive function to calculate the sum of the digits of a number
int sum_of_digits(int n) {
    // Base case: if the number is 0, the sum is 0
    if (n == 0) {
        return 0;
    } else {
        // Add the last digit (n % 10) to the sum of the digits of the rest of the number (n / 10)
        return (n % 10) + sum_of_digits(n / 10);
    }
}

int main() {
    int number;

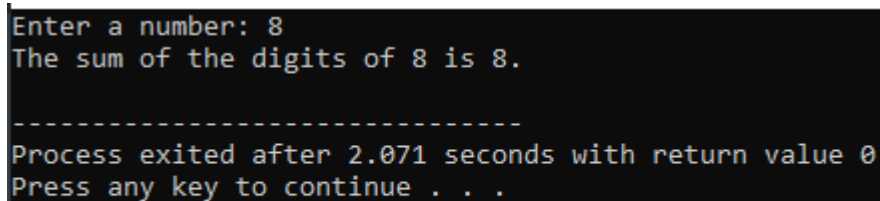
    // Get the number from the user
    printf("Enter a number: ");
    scanf("%d", &number);

    // Calculate the sum of the digits using the recursive function
    int sum = sum_of_digits(number);

    // Print the result
    printf("The sum of the digits of %d is %d.\n", number, sum);

    return 0;
}
```

Output:-

A screenshot of a terminal window showing the execution of the C program. The user enters the number 8. The program outputs "The sum of the digits of 8 is 8." followed by a dashed line separator. Below the separator, it shows "Process exited after 2.071 seconds with return value 0" and "Press any key to continue . . .".

```
Enter a number: 8
The sum of the digits of 8 is 8.
-----
Process exited after 2.071 seconds with return value 0
Press any key to continue . . .
```

6. Write a c program to read an integer number and print the reverse of that number using recursion.

Code:-

```
#include <stdio.h>

// Function prototype
void printReverse(int num);

int main() {
    int number;

    // Read an integer from the user
    printf("Enter an integer: ");
    scanf("%d", &number);

    // Handle negative numbers by removing the sign for the reversal
    if (number < 0) {
        printf("-");
        number = -number;
    }

    // Print the reversed number
    printf("Reversed number: ");
    printReverse(number);
    printf("\n");

    return 0;
}

// Recursive function to print the reverse of the number
void printReverse(int num) {
    if (num == 0) {
        return;
    }

    // Extract the last digit and print the reverse of the remaining digits
    printf("%d", num % 10);
    printReverse(num / 10);
}
```


Output:-

```
Enter an integer: 213  
Reversed number: 312
```

```
-----  
Process exited after 7.974 seconds with return value 0  
Press any key to continue . . .
```

7. Using function, Write a c program to find minimum and maximum between two numbers.

Code :-

```
#include <stdio.h>
// Function to find the minimum of two numbers
int findMinimum(int a, int b) {
    return (a < b) ? a : b;
}

// Function to find the maximum of two numbers
int findMaximum(int a, int b) {
    return (a > b) ? a : b;
}

int main() {
    int num1, num2;

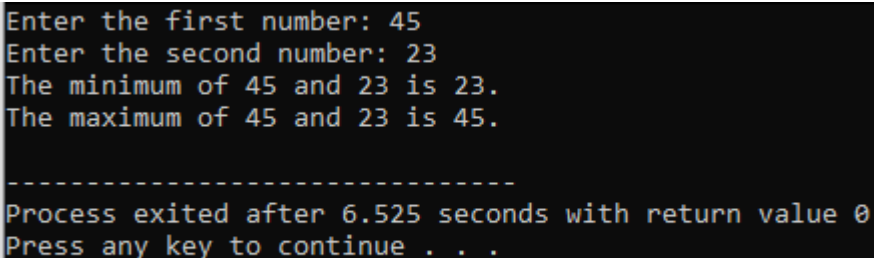
    // Prompt user for input
    printf("Enter the first number: ");
    scanf("%d", &num1);

    printf("Enter the second number: ");
    scanf("%d", &num2);

    // Find and display the minimum and maximum
    printf("The minimum of %d and %d is %d.\n", num1, num2, findMinimum(num1, num2));
    printf("The maximum of %d and %d is %d.\n", num1, num2, findMaximum(num1, num2));

    return 0;
}
```

Output :-

A screenshot of a terminal window showing the execution of a C program. The program prompts the user to enter two numbers, 45 and 23. It then displays the minimum (23) and maximum (45) of these numbers. The output is shown in a monospaced font on a dark background.

```
Enter the first number: 45
Enter the second number: 23
The minimum of 45 and 23 is 23.
The maximum of 45 and 23 is 45.

-----
Process exited after 6.525 seconds with return value 0
Press any key to continue . . .
```

8. Write a c program to check whether a number is even or odd using function.

Code :-

```
#include <stdio.h>
// Function to check if a number is even or odd
void checkEvenOdd(int num) {
    if (num % 2 == 0) {
        printf("%d is an Even number.\n", num);
    } else {
        printf("%d is an Odd number.\n", num);
    }
}

int main() {
    int number;

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

    // Call the function to check if the number is even or odd
    checkEvenOdd(number);

    return 0;
}
```

Output :-

```
Enter a number: 4
4 is an Even number.

-----
Process exited after 2.678 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number: 7
7 is an Odd number.

-----
Process exited after 3.451 seconds with return value 0
Press any key to continue . . .
```

9. Write a c program to check whether a number is prime ,Armstrong or perfect number using function.

Code :-

```
#include <stdio.h>
#include <math.h>
int isPrime(int num) {
    if (num <= 1) return 0;
    if (num == 2) return 1;
    if (num % 2 == 0) return 0;

    for (int i = 3; i <= sqrt(num); i += 2) {
        if (num % i == 0) return 0;
    }
    return 1;
}
int isArmstrong(int num) {
    int original = num, sum = 0, digit, n = 0;
    while (original != 0) {
        original /= 10;
        ++n;
    }

    original = num;
    while (original != 0) {
        digit = original % 10;
        sum += pow(digit, n);
        original /= 10;
    }

    return (num == sum);
}
int isPerfect(int num) {
    int sum = 0;

    for (int i = 1; i <= num / 2; ++i) {
        if (num % i == 0) {
            sum += i;
        }
    }

    return (num == sum);
}

int main() {
    int number;
```

```

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

if (isPrime(number)) {
    printf("%d is a Prime number.\n", number);
} else {
    printf("%d is not a Prime number.\n", number);
}

if (isArmstrong(number)) {
    printf("%d is an Armstrong number.\n", number);
} else {
    printf("%d is not an Armstrong number.\n", number);
}

if (isPerfect(number)) {
    printf("%d is a Perfect number.\n", number);
} else {
    printf("%d is not a Perfect number.\n", number);
}

return 0;
}

```

OUTPUT :-

```

Enter a number: 5
5 is a Prime number.
5 is an Armstrong number.
5 is not a Perfect number.

-----
Process exited after 4.541 seconds with return value 0
Press any key to continue . . .

```

```

Enter a number: 153
153 is not a Prime number.
153 is an Armstrong number.
153 is not a Perfect number.

-----
Process exited after 2.99 seconds with return value 0
Press any key to continue . . .

```

10. Write a c program to find the power of any number using recursion.

Code :-

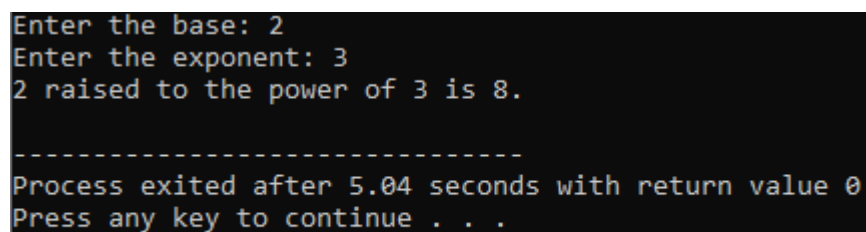
```
#include <stdio.h>

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

int main() {
    int base, exponent;
    printf("Enter the base: ");
    scanf("%d", &base);
    printf("Enter the exponent: ");
    scanf("%d", &exponent);
    if (exponent < 0) {
        printf("Exponent should be a non-negative integer.\n");
    } else {
        int result = power(base, exponent);
        printf("%d raised to the power of %d is %d.\n", base, exponent, result);
    }

    return 0;
}
```

Output :-

A screenshot of a terminal window showing the execution of the C program. The user enters '2' for the base and '3' for the exponent. The program outputs '2 raised to the power of 3 is 8.' followed by a separator line. Below the separator, it shows 'Process exited after 5.04 seconds with return value 0' and 'Press any key to continue . . .'.

```
Enter the base: 2
Enter the exponent: 3
2 raised to the power of 3 is 8.

-----
Process exited after 5.04 seconds with return value 0
Press any key to continue . . .
```