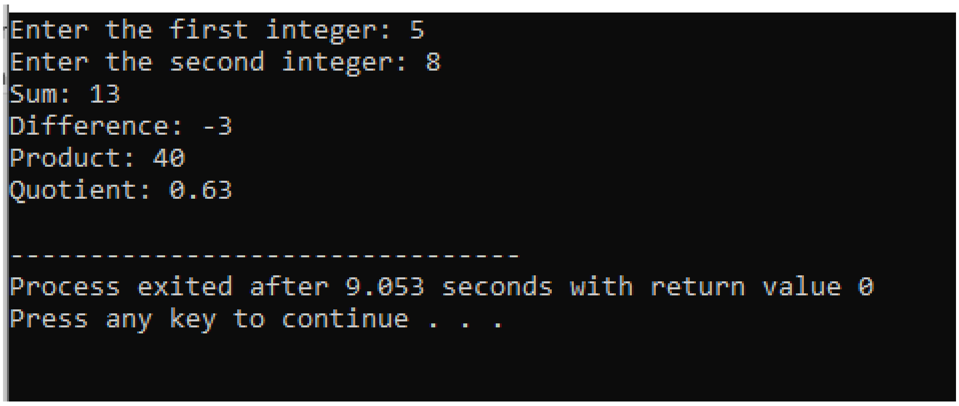
1. **Write a c program to add,subtract,multiply and divided two integers using a user defined type function with a return type.**

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:-



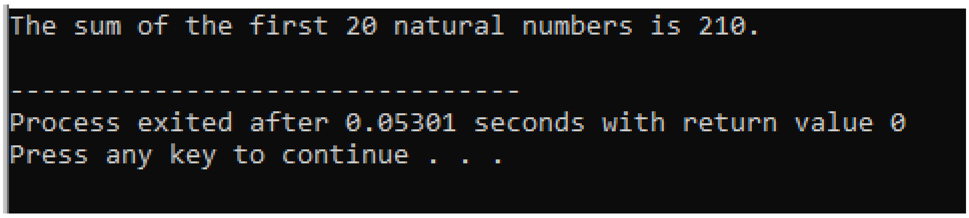
```
Enter the first integer: 5
Enter the second integer: 8
Sum: 13
Difference: -3
Product: 40
Quotient: 0.63

--------------------------------
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:-



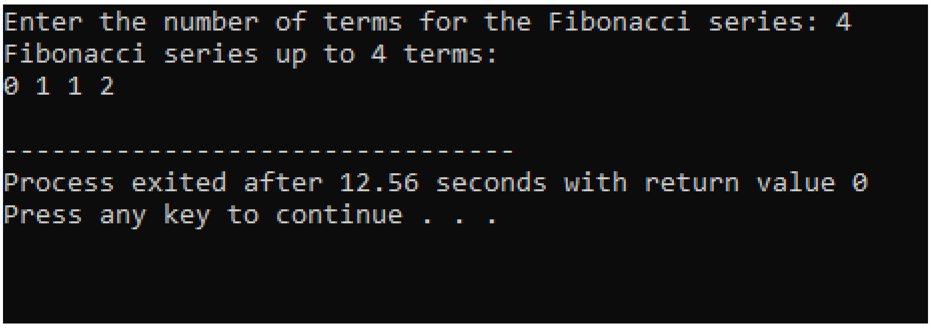
```
The sum of the first 20 natural numbers is 210.

--------------------------------
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:-



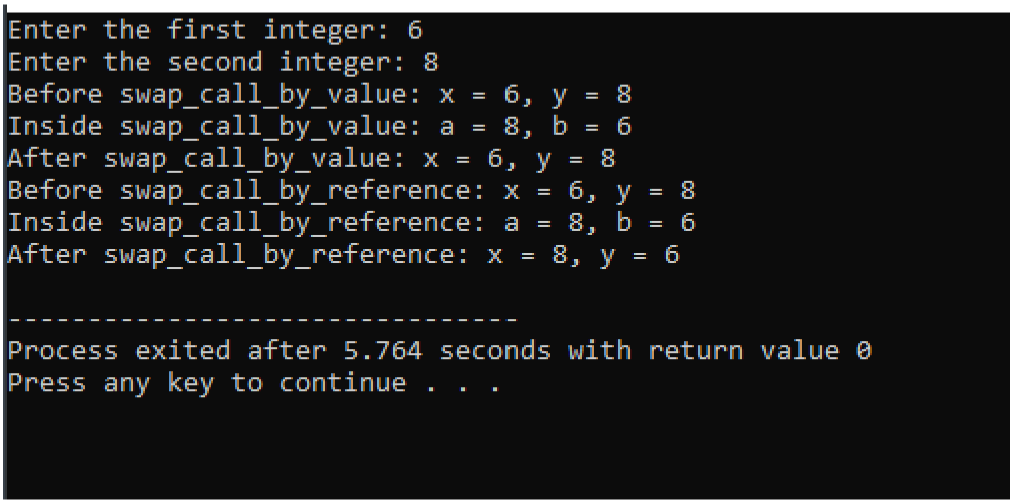
```
Enter the number of terms for the Fibonacci series: 4
Fibonacci series up to 4 terms:
0 1 1 2

--------------------------------
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-referencemethods 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:-



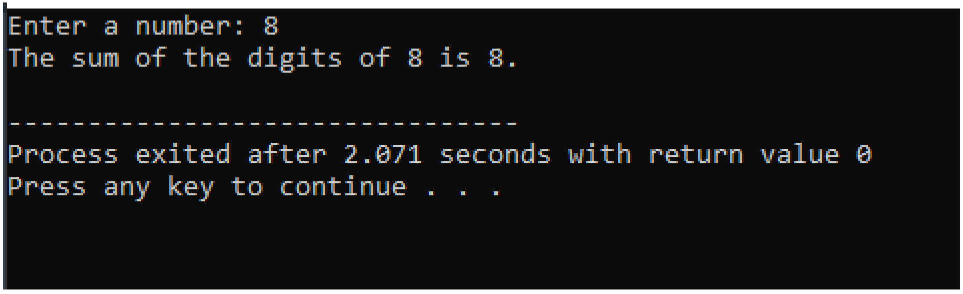
```
Enter the first integer: 6
Enter the second integer: 8
Before swap_call_by_value: x = 6, y = 8
Inside swap_call_by_value: a = 8, b = 6
After swap_call_by_value: x = 6, y = 8
Before swap_call_by_reference: x = 6, y = 8
Inside swap_call_by_reference: a = 8, b = 6
After swap_call_by_reference: x = 8,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



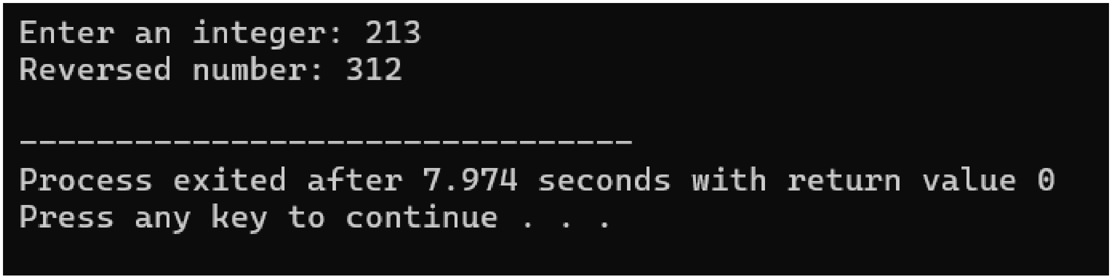
```
Enter a number: 8
The sum of the digits of 8 is 8.

--------------------------------
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:-



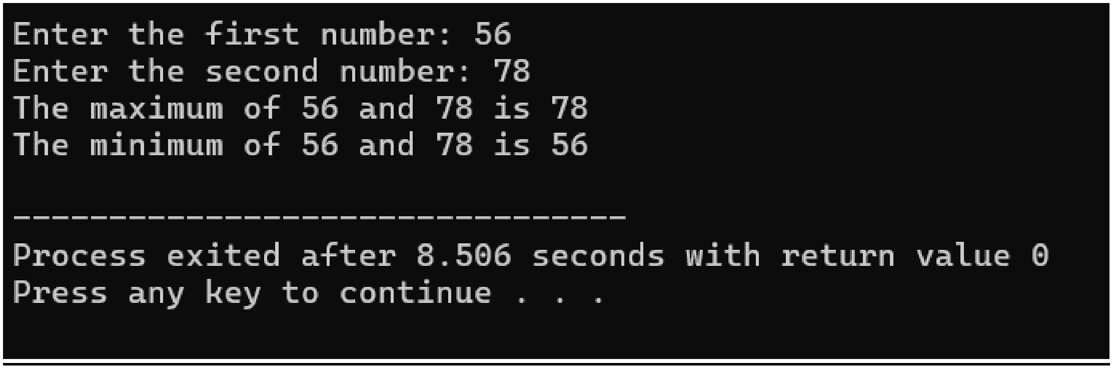
```
Enter an integer: 213
Reversed number: 312

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

7. **Using functions, write a C program to find the maximum and minimum between two numbers.**

Code:-

```
  #include <stdio.h>

// Function prototypes
int findMax(int a, int b);
int findMin(int a, int b);

int main() {
   int num1, num2;

   // Read two integers from the user
   printf("Enter the first number: ");
   scanf("%d", &num1);

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

   // Find the maximum and minimum
   int max = findMax(num1, num2);
   int min = findMin(num1, num2);

   // Print the results
   printf("The maximum of %d and %d is %d\n", num1, num2, max);
   printf("The minimum of %d and %d is %d\n", num1, num2, min);

   return 0;
}

// Function to find the maximum of two numbers
int findMax(int a, int b) {
   if (a > b) {
      return a;
   } else {
      return b;
   }
}

// Function to find the minimum of two numbers
int findMin(int a, int b) {
   if (a < b) {
      return a;
   } else {
      return b;
   }
}
```

Output:-



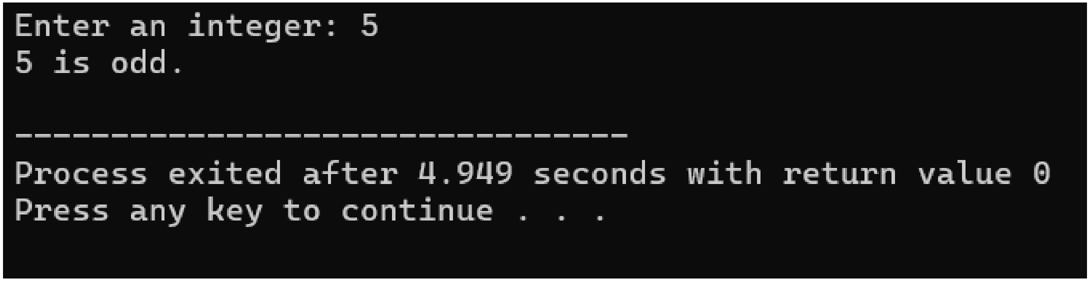
```
Enter the first number: 56
Enter the second number: 78
The maximum of 56 and 78 is 78
The minimum of 56 and 78 is 56

--------------------------------
Process exited after 8.506 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 functions.

Code:-

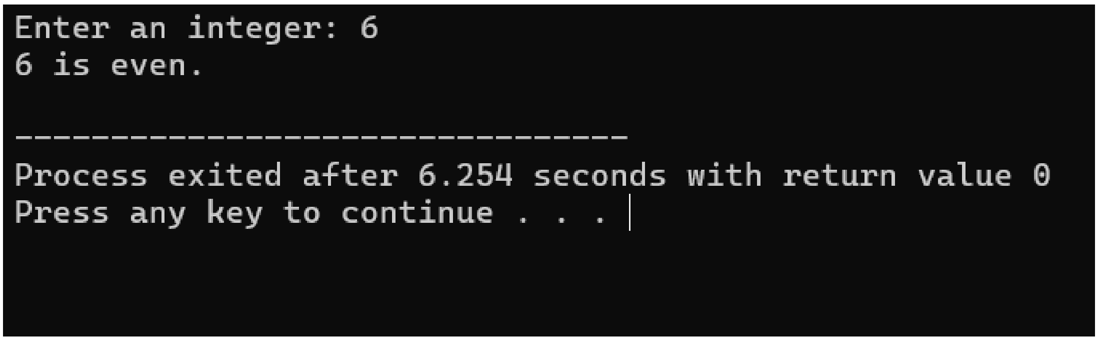
```
    #include <stdio.h>

// Function prototypes
int isEven(int num);
int isOdd(int num);

int main() {
   int number;

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

   // Check if the number is even or odd
   if (isEven(number)) {
      printf("%d is even.\n", number);
   } else if (isOdd(number)) {
      printf("%d is odd.\n", number);
   } else {
      printf("Error: Invalid input.\n");
   }

   return 0;
}

// Function to check if a number is even
int isEven(int num) {
   return (num % 2 == 0);
}

// Function to check if a number is odd
int isOdd(int num) {
   return (num % 2 != 0);
}
```

Output:-



```
Enter an integer: 5
5 is odd.

--------------------------------
Process exited after 4.949 seconds with return value 0
Press any key to continue . . .
```

-



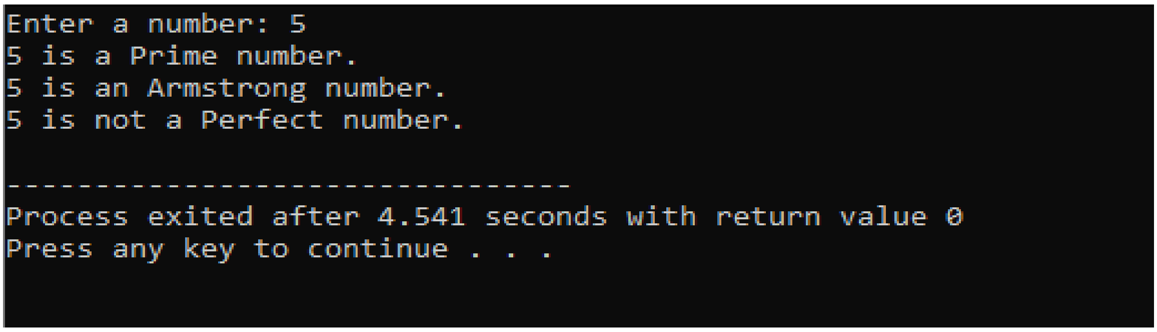
```
Enter an integer: 6
6 is even.

--------------------------------
Process exited after 6.254 seconds with return value 0
Press any key to continue . . .
```

9. **Write a C program to check whether a number is a prime, Armstrong, or Perfect number using functions.**

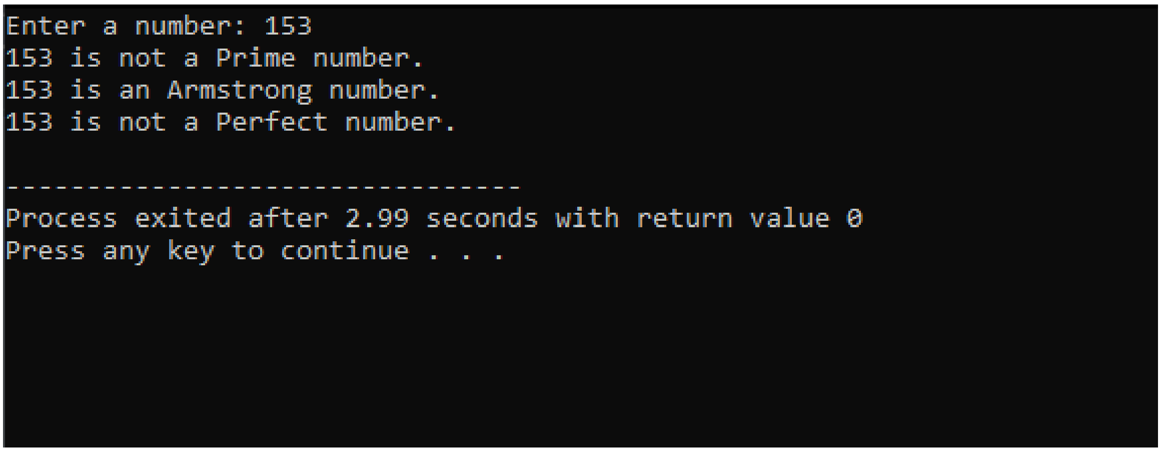
Code:-

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

// Function prototypes
int isPrime(int num);
int isArmstrong(int num);
int isPerfect(int num);

int main() {
  int number;

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

  // Check if the number is prime
  if (isPrime(number)) {
    printf("%d is a prime number.\n", number);
  } else if (isArmstrong(number)) {
    printf("%d is an Armstrong number.\n", number);
  } else if (isPerfect(number)) {
    printf("%d is a perfect number.\n", number);
  } else {
    printf("%d is neither a prime, Armstrong, nor perfect number.\n", number);
  }

  return 0;
}

// Function to check if a number is prime
int isPrime(int num) {
  if (num <= 1) return 0; // 0 and 1 are not prime numbers
  if (num == 2) return 1; // 2 is a prime number
  if (num % 2 == 0) return 0; // All other even numbers are not prime

  for (int i = 3; i <= sqrt(num); i += 2) {
    if (num % i == 0) return 0;
  }

  return 1;
}

// Function to check if a number is an Armstrong number
int isArmstrong(int num) {
  int originalNum = num;
  int sum = 0;
  int digits = 0;

  // Count the number of digits
  while (num != 0) {
    num /= 10;
    digits++;
  }

  num = originalNum;

  // Calculate the sum of powers of digits
  while (num != 0) {
    int digit = num % 10;
    sum += pow(digit, digits);
    num /= 10;
  }

  return (sum == originalNum);
}

// Function to check if a number is perfect
int isPerfect(int num) {
  if (num <= 1) return 0; // 1 is not a perfect number

  int sum = 0;

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

  return (sum == num);
}
```

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 . . .
```



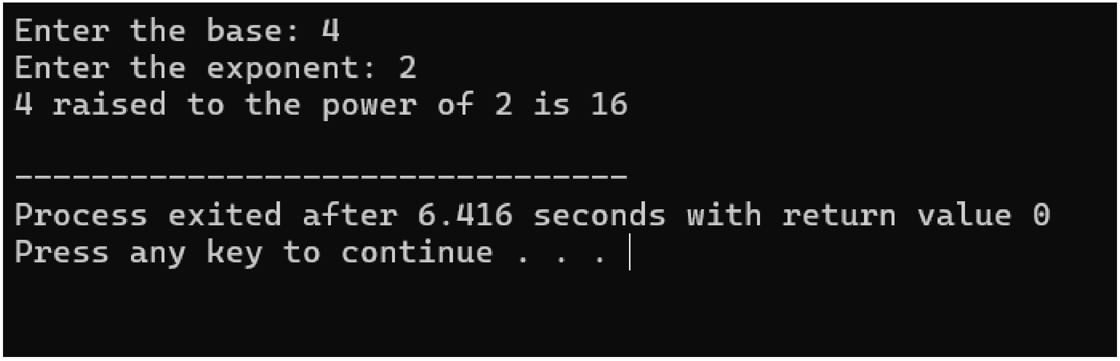
```
Enter a number: 153
153 is not a Prime number.
153 is an Armstrong number.
153 is not a Perfect number.

--------------------------------
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>

// Function prototype
int power(int base, int exponent);

int main() {
    int base, exponent;

    // Read the base and exponent from the user
    printf("Enter the base: ");
    scanf("%d", &base);

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

    // Calculate the power using the recursive function
    int result = power(base, exponent);

    // Print the result
    printf("%d raised to the power of %d is %d\n", base, exponent, result);

    return 0;
}

// Recursive function to calculate the power of a number
int power(int base, int exponent) {
    // Base case: any number raised to the power of 0 is 1
    if (exponent == 0) {
        return 1;
    }
    // Recursive case: multiply base with the result of power(base, exponent - 1)
    return base * power(base, exponent - 1);
}
```

Output:-



```
Enter the base: 4
Enter the exponent: 2
4 raised to the power of 2 is 16

--------------------------------
Process exited after 6.416 seconds with return value 0
Press any key to continue . . .
```