|  |  |  |  |
| --- | --- | --- | --- |
| Sl.  No. | **C PROGRAMMING** | Page  No. | Teacher’s  Signature |
|  | **FUNCTION** |  |  |
| 1 | Write a C program to find cube of any number using function. |  |  |
| 2 | Write a C program to find diameter, circumference and area of circle using functions. |  |  |
| 3 | Write a C program to find maximum and minimum between two numbers using functions. |  |  |
| 4 | Write a C program to check whether a number is even or odd using functions. |  |  |
| 5 | Write a C program to check whether a number is prime or not using function. |  |  |
| 6 | Write a program to check whether a number is an Armstrong number or not using function. |  |  |
| 7 | WAP to check a number is perfect number or not using functions. |  |  |
| 8 | Write a C program to find all prime numbers between given interval using functions. |  |  |
| 9 | Write a C program to print all strong numbers between given interval using functions. |  |  |
| 10 | Write a C program to print all Armstrong numbers between given interval using functions. |  |  |
| 11 | Write a C program to print all perfect numbers between given interval using functions. |  |  |
|  | **RECURSION** |  |  |
| 1 | Write a C program to find power of any number using recursion. |  |  |
| 2 | Write a C program to print all natural numbers between 1 to n using recursion. |  |  |
| 3 | Write a C program to print all even or odd numbers in given range using recursion. |  |  |
| 4 | Write a C program to find sum of all natural numbers between 1 to n using recursion. |  |  |
| 5 | Write a C program to find sum of all even or odd numbers in given range using recursion. |  |  |
| 6 | Write a C program to find reverse of any number using recursion. |  |  |
| 7 | Write a C program to check whether a number is palindrome or not using recursion. |  |  |
| 8 | Write a C program to find sum of digits of a given number using recursion. |  |  |
| 9 | Write a C program to find factorial of any number using recursion. |  |  |
| 10 | Write a C program to generate nth Fibonacci term using recursion. |  |  |
| 11 | Write a C program to find GCD (HCF) of two numbers using recursion. |  |  |
| 12 | Write a C program to find LCM of two numbers using recursion. |  |  |
| 13 | Write a C program to display all array elements using recursion. |  |  |
| 14 | Write a C program to find sum of elements of array using recursion. |  |  |
| 15 | Write a C program to find maximum and minimum elements in array using recursion. |  |  |

--------------------**CONTENTS**---------------------

* Write a C program to find cube of any number using function.

C CODE:

#include <stdio.h>

int cube(int n);

int main()

{

    int num;

    printf("Enter the number:\n");

    scanf("%d", &num);

    printf("cube of %d is %d.", num, cube(num));

    return 0;

}

int cube(int n)

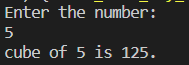
{

    int result;

    result = n \* n \* n;

    return (result);

}

OUTPUT:  


* Write a C program to find diameter, circumference and area of circle using functions.

C CODE:

#include <stdio.h>

void circle(float r);

int main()

{

    float R;

    printf("Enter the radius of circle:\n");

    scanf("%f", &R);

    circle(R);

    return 0;

}

void circle(float r)

{

    float C, D, A;

    D = 2 \* r;

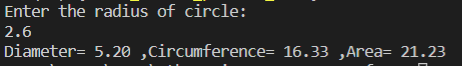
    C = 2 \* 3.14 \* r;

    A = 3.14 \* r \* r;

    printf("Diameter= %.2f ,Circumference= %.2f ,Area= %.2f", D, C, A);

}

OUTPUT:



* Write a C program to find maximum and minimum between two numbers using functions.

C CODE:

#include <stdio.h>

void max(int num1, int num2)

{

    if (num1 > num2)

    {

        printf("%d is greater than %d.", num1, num2);

    }

    else if (num2 > num1)

    {

        printf("%d is greater than %d.", num2, num1);

    }

    else

    {

        printf("Eror!");

    }

}

int main()

{

    int num1, num2;

    printf("Enter two number respectively:\t");

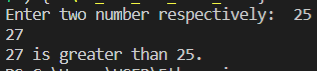
    scanf("%d%d", &num1, &num2);

    max(num1, num2);

    return 0;

}

OUTPUT:



* Write a C program to check whether a number is even or odd using functions.

C CODE:

#include <stdio.h>

void even\_or\_odd(int num);

int main()

{

    int n;

    printf("Enter a number:\n");

    scanf("%d", &n);

    even\_or\_odd(n);

    return 0;

}

void even\_or\_odd(int num)

{

    if (num % 2 == 0)

    {

        printf("%d is a even number.", num);

    }

    else

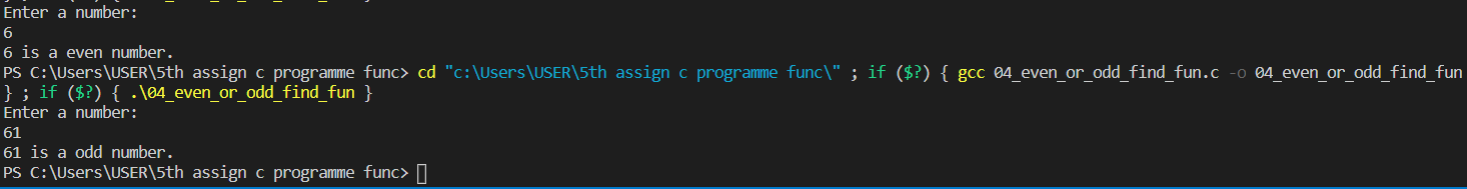
    {

        printf("%d is a odd number.", num);

    }

}

OUTPUT:



* Write a C program to check whether a number is prime or not using function.

C CODE:

#include <stdio.h>

void check\_prime(int num);

int main()

{

    int N;

    printf("Enter a number:\n");

    scanf("%d", &N);

    check\_prime(N);

    return 0;

}

void check\_prime(int num)

{

    int flag = 1;

    for (int i = 2; i <= num / 2; i++)

    {

        if (num % i == 0)

        {

            printf("This is not a prime number.\n");

            flag = 0;

            break;

        }

    }

    if (flag)

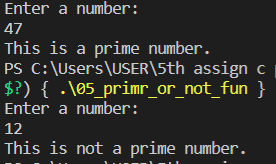
    {

        printf("This is a prime number.\n");

    }

}

OUTPUT:



* Write a program to check whether a number is an Armstrong number or not using function.

C CODE:

#include <stdio.h>

void check\_angstrom(int n);

int main()

{

    int num, digit;

    printf("Enter a number:\n");

    scanf("%d", &num);

    check\_angstrom(num);

    return 0;

}

void check\_angstrom(int n)

{

    int result, rem, sum = 0, temp;

    temp = n;

    while (temp != 0)

    {

        rem = temp % 10;

        sum = sum + (rem \* rem \* rem);

        temp = temp / 10;

    }

    if (sum == n)

    {

        printf("Angstrom Number.\n");

    }

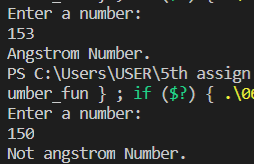
    else

    {

        printf("Not angstrom Number.\n");

    }

}

OUTPUT:  


* WAP to check a number is perfect number or not using functions.

C CODE:

#include <stdio.h>

void check\_perfect(int x);

int main()

{

    int num;

    printf("Enter a number:\n");

    scanf("%d", &num);

    check\_perfect(num);

    return 0;

}

void check\_perfect(int x)

{

    int i, sum, temp;

    temp = x;

    sum = 0;

    for (i = 1; i < temp; i++)

    {

        if (temp % i == 0)

        {

            sum = sum + i;

        }

    }

    if (sum == x)

    {

        printf("This is perfect number.\n");

    }

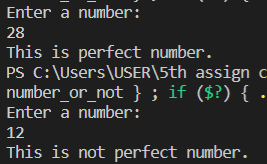
    else

    {

        printf("This is not perfect number.\n");

    }

}

OUTPUT:  


* Write a C program to find all prime numbers between given interval using functions.

C CODE:

#include <stdio.h>

void count\_prime(int N);

int main()

{

    int num, num1, num2;

    printf("Enter the range:\n");

    scanf("%d%d", &num1, &num2);

    printf("All prime numbers are:\n");

    for (int j = num1; j <= num2; j++)

    {

        count\_prime(j);

    }

    return 0;

}

void count\_prime(int N)

{

    int count = 0;

    for (int i = 1; i <= N; i++)

    {

        if (N % i == 0)

        {

            count++;

        }

    }

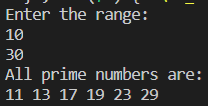
    if (count == 2)

    {

        printf("%d ", N);

    }

}

OUTPUT:  


* Write a C program to print all strong numbers between given interval using functions.

C CODE:

#include <stdio.h>

void check\_strong(int num);

int main()

{

    int num1, num2;

    printf("Enter a range:\n");

    scanf("%d%d", &num1, &num2);

    printf("Strong numbers are between range:\n");

    for (size\_t i = num1; i <= num2; i++)

    {

        check\_strong(i);

    }

    return 0;

}

void check\_strong(int num)

{

    int rem, temp, sum = 0, fact;

    temp = num;

    while (temp != 0)

    {

        rem = temp % 10;

        fact = 1;

        for (int i = 1; i <= rem; i++)

        {

            fact = fact \* i;

        }

        sum = sum + fact;

        temp = temp / 10;

    }

    if (sum == num)

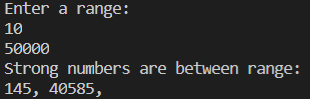
    {

        printf("%d, ", num);

    }

}

OUTPUT:



* Write a C program to print all Armstrong numbers between given interval using functions.

C CODE:

#include <stdio.h>

void armstrong(int x);

int main()

{

    int n1, n2;

    printf("Enter the lower and upper limit respectively:\n");

    scanf("%d%d", &n1, &n2);

    printf("Armstrong numbers are:\n");

    for (int i = n1; i < n2; i++)

    {

        armstrong(i);

    }

    return 0;

}

void armstrong(int x)

{

    int rem, sum = 0, temp;

    temp = x;

    while (temp != 0)

    {

        rem = temp % 10;

        sum = sum + (rem \* rem \* rem);

        temp = temp / 10;

    }

    if (sum == x)

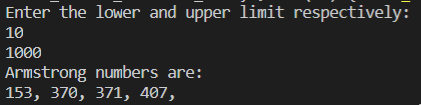
    {

        printf("%d, ", x);

    }

}

OUTPUT:



* Write a C program to print all perfect numbers between given interval using functions.

C CODE:

#include <stdio.h>

void check\_perfect(int, int);

int main()

{

    int n1, n2;

    printf("Enter lower limit and upper limit respectively:\n");

    scanf("%d%d", &n1, &n2);

    check\_perfect(n1, n2);

    return 0;

}

void check\_perfect(int x, int y)

{

    int i, j, sum;

    printf("Perfect numbers are:\n");

    for (i = x; i <= y; i++)

    {

        sum = 0;

        for (j = 1; j < i; j++)

        {

            if (i % j == 0)

            {

                sum = sum + j;

            }

        }

        if (sum == i)

        {

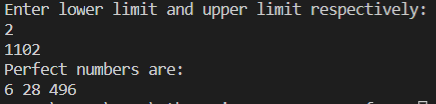
            printf("%d ", i);

        }

    }

}

OUTPUT:



**Recursion**

* Write a C program to find power of any number using recursion**.**

C CODE:

#include <stdio.h>

int power(int x, int y);

int main()

{

    int n1, n2;

    printf("Enter base and power respectively:\n");

    scanf("%d%d", &n1, &n2);

    printf("%d^%d=%d", n1, n2, power(n1, n2));

    return 0;

}

int power(int x, int y)

{

    if (y == 0)

    {

        return (1);

    }

    else

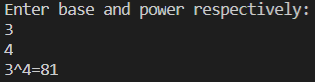
    {

        return (x \* power(x, (y - 1)));

    }

}

OUTPUT:

****

* Write a C program to print all natural numbers between 1 to n using recursion.

C CODE:

#include <stdio.h>

void print\_natural\_number(int x);

int main()

{

    int n;

    printf("Enter upper limit: ");

    scanf("%d", &n);

    print\_natural\_number(n);

    return 0;

}

void print\_natural\_number(int x)

{

    if (x == 0)

    {

        return;

    }

    else

    {

        (print\_natural\_number(x - 1));

    }

    printf(" %d", x);

}

OUTPUT:  


* Write a C program to print all even or odd numbers in given range using recursion.

C CODE:

#include <stdio.h>

void print\_even\_number(int x);

void print\_odd\_number(int x);

int main()

{

    int n;

    printf("Enter upper limit: ");

    scanf("%d", &n);

    printf("Even Number: ");

    print\_even\_number(n);

    printf("\nOdd Number: ");

    print\_odd\_number(n);

    return 0;

}

void print\_even\_number(int x)

{

    if (x == 0)

    {

        return;

    }

    else

    {

        print\_even\_number(x - 1);

    }

    if (x % 2 == 0)

    {

        printf("%d ", x);

    }

}

void print\_odd\_number(int x)

{

    if (x == 0)

    {

        return;

    }

    else

    {

        print\_odd\_number(x - 1);

    }

    if (x % 2 != 0)

    {

        printf("%d ", x);

    }

}

OUTPUT:



* Write a C program to find sum of all natural numbers between 1 to n using recursion.

C CODE:

#include <stdio.h>

int sum(int x);

int main()

{

    int n;

    printf("Enter upper limit:\n");

    scanf("%d", &n);

    printf("Summation= %d", sum(n));

    return 0;

}

int sum(int x)

{

    if (x == 1)

    {

        return (1);

    }

    else

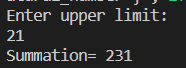
    {

        return (x + sum(x - 1));

    }

}

OUTPUT:



* Write a C program to find sum of all even or odd numbers in given range using recursion.

C CODE:

* Write a C program to find reverse of any number using recursion.

C CODE:

#include <stdio.h>

int reverse(int x);

int sum = 0;

int main()

{

    int num;

    printf("Enter a number:\n");

    scanf("%d", &num);

    printf("Reverse: %d", reverse(num));

    return 0;

}

int reverse(int x)

{

    if (x)

    {

        sum = sum \* 10 + (x % 10);

        reverse(x / 10);

    }

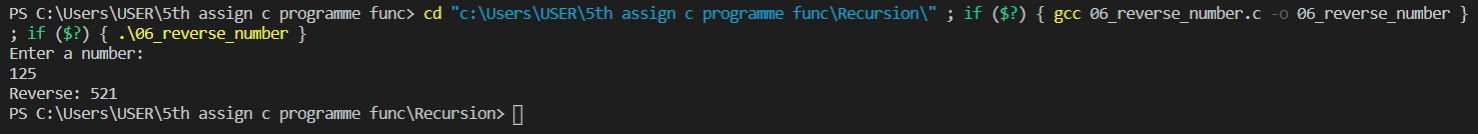
    else

    {

        return (sum);

    }

}

OUTPUT:  


* Write a C program to check whether a number is palindrome or not using recursion.

C CODE:

#include <stdio.h>

int is\_pal(int x);

int sum = 0;

int main()

{

    int num, palindrome;

    printf("Enter a number:\n");

    scanf("%d", &num);

    palindrome = is\_pal(num);

    if (palindrome == num)

    {

        printf("This is a palindrome number.\n");

    }

    else

    {

        printf("This is not a palindrome number.\n");

    }

    return 0;

}

int is\_pal(int x)

{

    if (x)

    {

        sum = sum \* 10 + (x % 10);

        is\_pal(x / 10);

    }

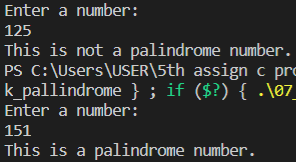
    else

    {

        return (sum);

    }

}

OUTPUT:  


* Write a C program to find sum of digits of a given number using recursion.

C CODE:

#include <stdio.h>

int sum\_digit(int x);

int sum = 0;

int main()

{

    int num;

    printf("Enter a number:\n");

    scanf("%d", &num);

    printf("Sum of digit of a given number: %d", sum\_digit(num));

    return 0;

}

int sum\_digit(int x)

{

    if (x)

    {

        sum = sum + (x % 10);

        sum\_digit(x / 10);

    }

    else

    {

        return (sum);

    }

}

OUTPUT:  


* Write a C program to find factorial of any number using recursion.

C CODE:  
#include <stdio.h>

int fact(int x);

int main()

{

    int num;

    printf("Enter a number: ");

    scanf("%d", &num);

    printf("%d! = %d", num, fact(num));

    return 0;

}

int fact(int x)

{

    if (x == 1)

    {

        return 1;

    }

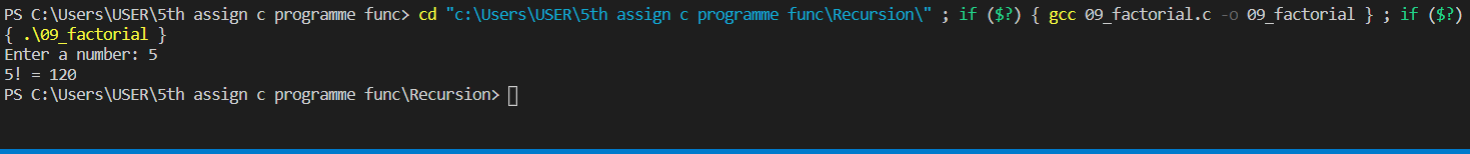
    else

    {

        return (x \* fact(x - 1));

    }

}

OUTPUT:  


* Write a C program to generate nth Fibonacci term using recursion.

C CODE:

#include <stdio.h>

int fabonacci(int x);

int main()

{

    int n;

    printf("Enter upper limit: ");

    scanf("%d", &n);

    printf("%dth fibonacci term is %d ",n,fabonacci(n));

    return 0;

}

int fabonacci(int x)

{

    if (x == 0)

    {

        return 0;

    }

    else if (x == 1)

    {

        return 1;

    }

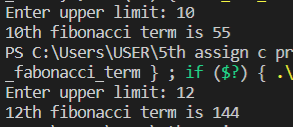
    else

    {

        return (fabonacci(x - 1) + fabonacci(x - 2));

    }

}

OUTPUT:  


* Write a C program to find GCD (HCF) of two numbers using recursion.

C CODE:

#include <stdio.h>

int gcd(int, int);

int main()

{

    int n1, n2;

    printf("Enter two number:\n");

    scanf("%d%d", &n1, &n2);

    printf("GCD of %d and %d is %d", n1, n2, gcd(n1, n2));

    return 0;

}

int gcd(int x, int y)

{

    if (x % y == 0)

    {

        return (y);

    }

    else

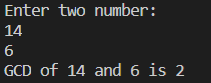
    {

        return (gcd(y, x % y));

    }

}

OUTPUT:



* Write a C program to find LCM of two numbers using recursion.

C CODE:

#include <stdio.h>

int gcd(int, int);

int lcm(int, int);

int main()

{

    int n1, n2;

    printf("Enter two number:\n");

    scanf("%d%d", &n1, &n2);

    printf("LCM of %d and %d is %d", n1, n2, lcm(n1, n2));

    return 0;

}

int gcd(int x, int y)

{

    if (x % y == 0)

    {

        return (y);

    }

    else

    {

        return (gcd(y, x % y));

    }

}

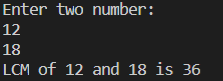
int lcm(int x, int y)

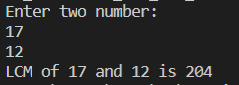
{

    return ((x / gcd(x, y)) \* y);

}

OUTPUT:





* Write a C program to display all array elements using recursion.

C CODE: