1. Write a C program to find power of a number using for loop.

#include <stdio.h>

// Function to calculate power

double power(double base, int exponent) {

double result = 1.0;

for (int i = 0; i < exponent; i++) {

result \*= base;

}

return result;

}

int main() {

double base;

int exponent;

// Input base and exponent from user

printf("Enter base: ");

scanf("%lf", &base);

printf("Enter exponent: ");

scanf("%d", &exponent);

// Calculate and print the result

printf("%.2lf ^ %d = %.2lf\n", base, exponent, power(base, exponent));

return 0;

}

1. Write a C program to find all factors of a number.

#include <stdio.h>

void printFactors(int number) {

printf("Factors of %d are: ", number);

for (int i = 1; i <= number; i++) {

if (number % i == 0) {

printf("%d ", i);

}

}

printf("\n");

}

int main() {

int number;

// Input number from user

printf("Enter a number: ");

scanf("%d", &number);

// Print factors

printFactors(number);

return 0;

}

1. Write a C program to calculate factorial of a number.

#include <stdio.h>

int factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int number;

// Input number from user

printf("Enter a number: ");

scanf("%d", &number);

// Calculate and print factorial

printf("Factorial of %d = %d\n", number, factorial(number));

return 0;

}

1. Write a C program to find HCF (GCD) of two numbers.

#include <stdio.h>

int findHCF(int a, int b) {

while (a != b) {

if (a > b) {

a -= b;

} else {

b -= a;

}

}

return a;

}

int main() {

int num1, num2;

// Input numbers from user

printf("Enter first number: ");

scanf("%d", &num1);

printf("Enter second number: ");

scanf("%d", &num2);

// Find and print HCF

printf("HCF of %d and %d = %d\n", num1, num2, findHCF(num1, num2));

return 0;

}

1. Write a C program to find LCM of two numbers.

#include <stdio.h>

int findLCM(int a, int b) {

int max = (a > b) ? a : b;

while (1) {

if (max % a == 0 && max % b == 0) {

return max;

}

++max;

}

}

int main() {

int num1, num2;

// Input numbers from user

printf("Enter first number: ");

scanf("%d", &num1);

printf("Enter second number: ");

scanf("%d", &num2);

// Find and print LCM

printf("LCM of %d and %d = %d\n", num1, num2, findLCM(num1, num2));

return 0;

}

1. Write a C program to check whether a number is Prime number or not.

#include <stdio.h>

int isPrime(int number) {

if (number <= 1) {

return 0; // Not prime

}

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return 0; // Not prime

}

}

return 1; // Prime

}

int main() {

int number;

// Input number from user

printf("Enter a number: ");

scanf("%d", &number);

// Check and print if the number is prime

if (isPrime(number)) {

printf("%d is a prime number.\n", number);

} else {

printf("%d is not a prime number.\n", number);

}

return 0;

}

1. Write a C program to print all Prime numbers between 1 to n.

#include <stdio.h>

int isPrime(int number) {

if (number <= 1) {

return 0; // Not prime

}

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return 0; // Not prime

}

}

return 1; // Prime

}

void printPrimes(int n) {

printf("Prime numbers between 1 and %d are: ", n);

for (int i = 2; i <= n; i++) {

if (isPrime(i)) {

printf("%d ", i);

}

}

printf("\n");

}

int main() {

int limit;

// Input limit from user

printf("Enter a limit: ");

scanf("%d", &limit);

// Print prime numbers up to the limit

printPrimes(limit);

return 0;

}

1. Write a C program to find sum of all prime numbers between 1 to n.

#include <stdio.h>

int isPrime(int number) {

if (number <= 1) {

return 0; // Not prime

}

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return 0; // Not prime

}

}

return 1; // Prime

}

int sumPrimes(int n) {

int sum = 0;

for (int i = 2; i <= n; i++) {

if (isPrime(i)) {

sum += i;

}

}

return sum;

}

int main() {

int limit;

// Input limit from user

printf("Enter a limit: ");

scanf("%d", &limit);

// Calculate and print the sum of prime numbers up to the limit

printf("Sum of prime numbers between 1 and %d = %d\n", limit, sumPrimes(limit));

return 0;

}

1. Write a C program to find all prime factors of a number.

#include <stdio.h>

void primeFactors(int number) {

printf("Prime factors of %d are: ", number);

for (int i = 2; i <= number; i++) {

while (number % i == 0) {

printf("%d ", i);

number /= i;

}

}

printf("\n");

}

int main() {

int num;

// Input number from user

printf("Enter a number: ");

scanf("%d", &num);

// Print prime factors

primeFactors(num);

return 0;

}

1. Write a C program to check whether a number is Armstrong number or not.

#include <stdio.h>

#include <math.h>

int isArmstrong(int number) {

int originalNumber, remainder, n = 0, result = 0;

originalNumber = number;

while (originalNumber != 0) {

originalNumber /= 10;

++n;

}

originalNumber = number;

while (originalNumber != 0) {

remainder = originalNumber % 10;

result += pow(remainder, n);

originalNumber /= 10;

}

return (result == number);

}

int main() {

int number;

// Input number from user

printf("Enter a number: ");

scanf("%d", &number);

// Check and print if the number is Armstrong

if (isArmstrong(number)) {

printf("%d is an Armstrong number.\n", number);

} else {

printf("%d is not an Armstrong number.\n", number);

}

return 0;

}

1. Write a C program to print all Armstrong numbers between 1 to n.

#include <stdio.h>

#include <math.h>

int isArmstrong(int number) {

int originalNumber, remainder, n = 0, result = 0;

originalNumber = number;

while (originalNumber != 0) {

originalNumber /= 10;

++n;

}

originalNumber = number;

while (originalNumber != 0) {

remainder = originalNumber % 10;

result += pow(remainder, n);

originalNumber /= 10;

}

return (result == number);

}

void printArmstrongNumbers(int n) {

printf("Armstrong numbers between 1 and %d are: ", n);

for (int i = 1; i <= n; i++) {

if (isArmstrong(i)) {

printf("%d ", i);

}

}

printf("\n");

}

int main() {

int limit;

// Input limit from user

printf("Enter a limit: ");

scanf("%d", &limit);

// Print Armstrong numbers up to the limit

printArmstrongNumbers(limit);

return 0;

}