// Function to calculate the factorial of a given number

fn factorial(n: u128) -> u128 {

if n == 0 {

return 1;

}

n \* factorial(n - 1)

}

// Function to calculate the Fibonacci sequence up to a given number

fn fibonacci(n: u128) -> u128 {

let mut a = 0;

let mut b = 1;

let mut c = 0;

for \_i in 0..n {

c = a + b;

a = b;

b = c;

}

c

}

// Function to calculate an arbitrary number of digits of pi using the Gauss-Legendre algorithm

fn pi\_digits(n: u128) -> f64 {

let mut pi: f64 = 0.0;

let mut a: f64 = 1.0;

let mut b: f64 = 1.0 / (2.0 as f64).sqrt();

let mut t: f64 = 1.0 / 4.0;

let mut p: f64 = 1.0;

for \_i in 0..n {

let prev\_a = a;

a = (a + b) / 2.0;

b = (prev\_a \* b).sqrt();

t -= p \* (a - prev\_a) \* (a - prev\_a);

p \*= 2.0;

pi = ((a + b) \* (a + b) / (4.0 \* t)).into();

}

pi

}

// Function to calculate a specific digit or sequence of digits of pi using the Chudnovsky algorithm

fn pi\_digit(n: u64) -> f64 {

let mut pi: f64 = 0.0;

let mut k: u64 = 0;

let mut m: u64 = 0;

let mut a: u64 = 0;

let mut b: u64 = 0;

let mut c: u64 = 0;

let mut d: u64 = 0;

let mut e: u64 = 0;

for \_i in 0..n {

k = m \* (4 \* m + 1);

a = (4 \* k) \* (4 \* m + 1);

b = (m \* (6 \* m + 1) + 1) \* (2 \* m + 1);

c = (4 \* m + 1) \* (2 \* m + 1);

d = (6 \* m + 2) \* (6 \* m + 2);

e = (4 \* m + 1) \* (6 \* m + 2);

pi = (a \* b) as f64 / (c \* d \* e) as f64;

m += 1;

}

pi

}

fn main() {

loop {

println!("Select a mathematical operation:");

println!("1. Factorial");

println!("2. Fibonacci sequence");

println!("3. Digits of pi");

println!("4. Specific digit or sequence of digits of pi (extra credit)");

println!("5. Exit");

let mut user\_input = String::new();

std::io::stdin().read\_line(&mut user\_input).unwrap();

let user\_input = user\_input.trim();

match user\_input {

"1" => {

println!("Enter a value for n:");

let mut n = String::new();

std::io::stdin().read\_line(&mut n).unwrap();

let n: u128 = n.trim().parse().unwrap();

let result = factorial(n);

println!("{}! = {}", n, result);

}

"2" => {

println!("Enter a value for n:");

let mut n = String::new();

std::io::stdin().read\_line(&mut n).unwrap();

let n: u128 = n.trim().parse().unwrap();

let result = fibonacci(n);

println!("Fibonacci number for n = {} is: {}", n, result);

let val = (3.0 / 2.0) \* (n as f64).powf(n as f64 - 1.0);

println!("(3/2)^(n-1) for n = {} is: {}", n, val);

}

"3" => {

println!("Enter a value for n (number of digits):");

let mut n = String::new();

std::io::stdin().read\_line(&mut n).unwrap();

let n: u128 = n.trim().parse().unwrap();

let result = pi\_digits(n);

println!("{} digits of pi = {}", n, result);

}

"4" => {

println!("Enter a value for n (specific digit):");

let mut n = String::new();

std::io::stdin().read\_line(&mut n).unwrap();

let n: u64 = n.trim().parse().unwrap();

let result = pi\_digit(n);

println!("{}th digit of pi = {}", n, result);

}

"5" => {

println!("Exiting program.");

break;

}

\_ => {

println!("Invalid input. Please enter a number from 1 to 5.");

}

}

}

}