Rust Lab 26th

Follow the instructions provided in each exercise. There'll be hints in some exercises; they will be spelled backward so that you don't accidentally go into the hints.

If your computer doesn't have the Rust compiler installed, you can visit the website https://play.rust-lang.org/ and complete the exercises there.

Fizz Buzz

For every number from 1 to 50 (inclusive), the program must print `FizzBuzz` if the number is both divisible by 3 and 5; or if the number is divisible by 5, the program must print `Buzz;` or if the number is divisible by 3, the program must print `Fizz`; otherwise, print the number itself.

Requirement

- You must use the loop in the program, either while-loop or for-loop.

Expected Output

```
1
2
Fizz
4
Buzz
Fizz
7
...
14
FizzBuzz
16
...
Fizz
49
Buzz
```

Multiplication Table

Write a program that prints out multiplication tables. The program should include tables 2 to 12 (inclusive). For each table, there should be 10 multiplications: from 1 to 10.

Requirement

- You must use the loop in the program, either while-loop or for-loop.

Expected Output

```
Table 2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
...
Table 12
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120
```

Printing Stair

Write a program that prints out the stair pattern.

Code Template

```
fn main() {
    let height = 4 /*can be changed*/;

    println!("height: {height}");

    // implement here
}
```

Expected Output (`height = 4` case)

```
height: 4
   *
  **
 ***
****
```

Expected Output (`height = 6` case)

```
height = 6
     *
    **
   ***
  ****
 *****
******
```

Hint

- `rebmun_enil - thgieh` slauqe sksiretsa eht fo tnorf ni secapsetihw fo rebmun eht ,enil hcae rof

---

Is Prime Number

Completes the function that determines whether the given unsigned integer is a prime number or not by returning a boolean value.

Code Template

```
fn is_prime(number: u32) -> bool {
      // complete it here
}

fn main() {
    for i in 1..=10 {
        let result = is_prime(i);
        println!("is {i} prime number: {result}");
    }
}
```

Expected Output

```
is 1 prime number: false
is 2 prime number: true
is 3 prime number: true
is 4 prime number: false
is 5 prime number: true
is 6 prime number: false
is 7 prime number: true
is 8 prime number: false
is 9 prime number: false
is 10 prime number: false
```

Hint

1. flesti rebmun eht dna eno si hcihw ,redniamer tuohtiw ti edivid nac srosivid owt ylno fi emirp si rebmun eht

---

Factorial

Completes a function that computes a factorial for the given number.

Requirement

- You must use recursive functions. Therefore, while and for loops are ***NOT ALLOWED.***

Code Template

```
fn factorial(n: u32) -> u32 {
      // complete it here
}

fn main() {
    let four = factorial(4);
    let zero = factorial(0);
    let one = factorial(1);

    println!("4! = {four}");
    println!("0! = {zero}");
    println!("1! = {one}");
}
```

Expected Output

```
4! = 24
0! = 1
1! = 1
```

Extra: Exponentiation with Recursion

Write a function that calculates the exponentiation by using the concept of recursion.

Requirement

- You must use recursive functions. Therefore, while and for loops are not allowed.

Code Template

```
fn exponent(base: f64, exp: u32) -> f64 {
      // complete it here
}

fn main() {
    let two_three = exponent(2.0, 3);
    let four_four = exponent(4.0, 4);
    let five_zero = exponent(5.0, 0);

    println!("2^3 = {two_three}");
    println!("4^4 = {four_four}");
    println!("5^0 = {five_zero}");
}
```

Expected Output

```
2^3 = 8
4^4 = 256
5^0 = 1
```

Hint

- ruof derewop evif semit evif ot slauqe evif derewop evif
