187. State Names and Abbreviations

```
fn filter_state_names(states: Vec<&str>, category: &str) -> Vec<String> {
    let full_state_names = [
        "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado",
"Connecticut",
        "Delaware", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois",
"Indiana", "Iowa",
        "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",
"Massachusetts", "Michigan",
        "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
"Nevada", "New Hampshire",
        "New Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma",
        "Oregon", "Pennsylvania", "Rhode Island", "South Carolina", "South
Dakota", "Tennessee",
        "Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia",
"Wisconsin", "Wyoming"
    ];

    states.into_iter().filter(|s| match category {
        "abb" => s.len() == 2,
        "full" => full_state_names.contains(&s),
        _ => false,
    }).map(|s| s.to_string()).collect()
}

fn main() {
    let states1 = vec!["Arizona", "CA", "NY", "Nevada"];
    let states2 = vec!["MT", "NJ", "TX", "ID", "IL"];

    println!("{:?}", filter_state_names(states1, "abb")); // ➞ ["CA", "NY"]
    println!("{:?}", filter_state_names(states1, "full")); // ➞ ["Arizona",
"Nevada"]
    println!("{:?}", filter_state_names(states2, "abb")); // ➞ ["MT", "NJ",
"TX", "ID", "IL"]
    println!("{:?}", filter_state_names(states2, "full")); // ➞ []
}
```

188. Functioninator 8000

```
fn inator_inator(word: &str) -> String {
    // Helper function to check if a character is a consonant
    fn is_consonant(c: char) -> bool {
        c.is_alphabetic() && !"aeiouAEIOU".contains(c)
    }

    let last_char = word.chars().last().unwrap_or(' '); // Get the last
character or default to a space

    // Determine the suffix based on the last character
    let suffix = if is_consonant(last_char) {
        "inator"
    } else {
        "-inator"
    };

    // Calculate the length and format it
    let length_suffix = format!("{}000", word.len());

    // Combine the parts into the final string
    format!("{}{} {}", word, suffix, length_suffix)
}

fn main() {
    let test_cases = ["Shrink", "Doom", "EvilClone"];

    for &word in test_cases.iter() {
        println!("{}", inator_inator(word));
    }
}

// Shrinkinator 6000
// Doominator 4000
// EvilClone-inator 9000
```

189. Pages in a Book

```rust
fn is_perfect_square(x: i64) -> bool {
    let s = (x as f64).sqrt() as i64;
    s * s == x
}

fn pages_in_book(total: i64) -> bool {
    if total <= 0 {
        return false;
    }

    let discriminant = 1 + 8 * total;
    if !is_perfect_square(discriminant) {
        return false;
    }

    let sqrt_disc = (discriminant as f64).sqrt() as i64;
    let n = (sqrt_disc - 1) / 2;

    // Verify that n is a positive integer and satisfies the original condition
    n > 0 && n * (n + 1) / 2 == total
}

fn main() {
    let test_cases = [5, 4005, 9453];

    for &total in test_cases.iter() {
        println!("Total pages {}: {}", total, pages_in_book(total));
    }
}

// Total pages 5: false
// Total pages 4005: true
// Total pages 9453: true
```

190. Highest Digit

```rust
fn highest_digit(n: u32) -> u32 {
    n.to_string()
        .chars()
        .filter_map(|c| c.to_digit(10))
        .max()
        .unwrap_or(0)
}

fn main() {
    let test_cases = [4666, 544, 379, 2, 377401];

    for &case in test_cases.iter() {
        println!("Highest digit in {}: {}", case, highest_digit(case));
    }
}

// Highest digit in 4666: 6
// Highest digit in 544: 5
// Highest digit in 379: 9
// Highest digit in 2: 2
// Highest digit in 377401: 7
```

191. Video Length in Seconds

```rust
fn minutes_to_seconds(video_length: &str) -> i32 {
    let parts: Vec<&str> = video_length.split(':').collect();

    if parts.len() != 2 {
        return -1; // Invalid format
    }

    let minutes: i32 = match parts[0].parse() {
        Ok(val) => val,
        Err(_) => return -1, // Invalid minutes value
    };

    let seconds: i32 = match parts[1].parse() {
        Ok(val) => val,
        Err(_) => return -1, // Invalid seconds value
    };

    if seconds >= 60 {
        return -1; // Invalid seconds value
    }

    minutes * 60 + seconds
}

fn main() {
    let test_cases = ["01:00", "13:56", "10:60", "121:49", "invalid"];

    for &case in test_cases.iter() {
        println!("Video Length '{}': {} seconds", case,
minutes_to_seconds(case));
    }
}

// Video Length '01:00': 60 seconds
// Video Length '13:56': 836 seconds
// Video Length '10:60': -1 seconds
// Video Length '121:49': 7309 seconds
// Video Length 'invalid': -1 seconds
```

192. Count Letters in a Word Search

```rust
fn letter_counter(grid: &[Vec<&str>], letter: &str) -> usize {
    grid.iter()
        .flat_map(|row| row.iter())
        .filter(|&&ch| ch == letter)
        .count()
}

fn main() {
    let grid = vec![
        vec!["D", "E", "Y", "H", "A", "D"],
        vec!["C", "B", "Z", "Y", "J", "K"],
        vec!["D", "B", "C", "A", "M", "N"],
        vec!["F", "G", "G", "R", "S", "R"],
        vec!["V", "X", "H", "A", "S", "S"],
    ];

    let letter = "D";
    println!("The letter '{}' appears {} times.", letter, letter_counter(&grid,
letter));

    let letter = "H";
    println!("The letter '{}' appears {} times.", letter, letter_counter(&grid,
letter));
}

// The letter 'D' appears 3 times.
// The letter 'H' appears 2 times.
```

193. Find the Other Two Side Lengths

```rust
const SQRT_3: f64 = 1.7320508075688772;

fn other_sides(shortest_side: f64) -> [f64; 2] {
    let hypotenuse = 2.0 * shortest_side;
    let other_side = shortest_side * SQRT_3;
    // Rounding to 2 decimal places
    [((hypotenuse * 100.0).round() / 100.0), ((other_side * 100.0).round() / 
100.0)]
}

fn main() {
    let test_cases = [1.0, 12.0, 2.0, 3.0];

    for &side in test_cases.iter() {
        let [longest, medium] = other_sides(side);
        println!("Shortest Side: {:.1}, Longest Side: {:.2}, Medium Side: 
{:.2}", side, longest, medium);
    }
}

// Shortest Side: 1.0, Longest Side: 2.00, Medium Side: 1.73
// Shortest Side: 12.0, Longest Side: 24.00, Medium Side: 20.78
// Shortest Side: 2.0, Longest Side: 4.00, Medium Side: 3.46
// Shortest Side: 3.0, Longest Side: 6.00, Medium Side: 5.20
```

194. War of Numbers

```rust
fn war_of_numbers(numbers: &[i32]) -> i32 {
    let (even_sum, odd_sum) = numbers.iter().fold((0, 0), |(even, odd), &num| {
        if num % 2 == 0 {
            (even + num, odd)
        } else {
            (even, odd + num)
        }
    });

    (even_sum - odd_sum).abs()
}

fn main() {
    let examples = [
        vec![2, 8, 7, 5],
        vec![12, 90, 75],
        vec![5, 9, 45, 6, 2, 7, 34, 8, 6, 90, 5, 243],
    ];

    for numbers in examples.iter() {
        println!("{}", war_of_numbers(numbers));
    }
}

// 2
// 27
// 168
```

195. Make a Circle with OOP

```rust
use std::f64::consts::PI;

struct Circle(f64);

impl Circle {
    fn new(radius: f64) -> Self { Self(radius) }
    fn get_area(&self) -> f64 { PI * self.0.powi(2) }
    fn get_perimeter(&self) -> f64 { 2.0 * PI * self.0 }
}

fn main() {
    let circles = [11.0, 4.44];
    for radius in circles {
        let circle = Circle::new(radius);
        println!("Radius: {:.2} - Area: {:.6}, Perimeter: {:.6}", radius,
circle.get_area(), circle.get_perimeter());
    }
}

// Radius: 11.00 - Area: 380.132711, Perimeter: 69.115038
// Radius: 4.44 - Area: 61.932101, Perimeter: 27.897343
```

196. Find the nth Tetrahedral Number

```rust
fn tetra(n: u64) -> u64 {
    n * (n + 1) * (n + 2) / 6
}

fn main() {
    println!("{}", tetra(2));
    println!("{}", tetra(5));
    println!("{}", tetra(6));
}

// 4
// 35
// 56
```

197. Joke Teller

```
use rand::seq::SliceRandom;
use std::io::{self, Write};

fn joke_teller_program() {
    let categories = ["knock-knock", "dad", "animal", "puns"];
    let (setup, punchline1, punchline2) = match categories.choose(&mut
rand::thread_rng()).unwrap() {
        &"knock-knock" => ("Knock, knock.", "Tank.", "You're welcome!"),
        &"dad" => ("Why did the scarecrow win an award?", "", "Because he was
outstanding in his field!"),
        &"animal" => ("Why don't scientists trust atoms?", "", "Because they
make up everything!"),
        _ => ("I used to be a baker because I kneaded dough.", "", ""),
    };

    if !setup.is_empty() { println!("{}", setup); }
    if !punchline1.is_empty() { let _ = prompt_user(punchline1); }
    println!("{}", punchline2);
}

fn prompt_user(prompt: &str) {
    print!("{} ", prompt);
    io::stdout().flush().unwrap();
    let _ = io::stdin().read_line(&mut String::new());
}

fn main() {
    joke_teller_program();
}

// Why did the scarecrow win an award?
// Because he was outstanding in his field!
```

198. Which Generation Are You?

```rust
fn generation(x: i32, y: char) -> &'static str {
    match (x, y) {
        (-3, 'm') => "great grandfather",
        (-3, 'f') => "great grandmother",
        (-2, 'm') => "grandfather",
        (-2, 'f') => "grandmother",
        (-1, 'm') => "father",
        (-1, 'f') => "mother",
        (0,  _)   => "me!",
        (1,  'm') => "son",
        (1,  'f') => "daughter",
        (2,  'm') => "grandson",
        (2,  'f') => "granddaughter",
        (3,  'm') => "great grandson",
        (3,  'f') => "great granddaughter",
        _ => "unknown"
    }
}

fn main() {
    println!("{}", generation(2, 'f'));
    println!("{}", generation(-3, 'm'));
    println!("{}", generation(1, 'f'));
}

// granddaughter
// great grandfather
// daughter
```

199. FizzBuzz Game

```rust
fn fizz_buzz_game() {
    println!("Welcome to the FizzBuzz Game!");
    println!("Let's play FizzBuzz!");

    for i in 1..=100 {
        let mut output = String::new();

        if i % 3 == 0 {
            output.push_str("Fizz");
        }

        if i % 5 == 0 {
            output.push_str("Buzz");
        }

        // Print the output or the number itself
        if output.is_empty() {
            println!("{}", i);
        } else {
            println!("{}", output);
        }
    }
}

fn main() {
    fizz_buzz_game();
}

// Welcome to the FizzBuzz Game!
// Let's play FizzBuzz!
// 1
// 2
// Fizz
// 4
// Buzz
// Fizz
```

200. Swap Pairs of Adjacent Digits

```rust
fn swap_pairs_of_adjacent_digits(number: u32) {
    let number_str = number.to_string();
    let length = number_str.len();

    // Check if the number has an even length
    if length % 2 != 0 {
        println!("Please enter an integer with an even length.");
        return;
    }

    // Swap pairs of adjacent digits
    let mut swapped_number = String::new();
    let chars: Vec<char> = number_str.chars().collect();

    for i in (0..length).step_by(2) {
        if i + 1 < length {
            swapped_number.push(chars[i + 1]);
            swapped_number.push(chars[i]);
        }
    }

    // Convert the result back to an integer
    let result: u32 = swapped_number.parse().unwrap();
    println!("Original Number: {}", number);
    println!("Number with Swapped Pairs of Adjacent Digits: {}", result);
}

fn main() {
    let number = 123456;
    swap_pairs_of_adjacent_digits(number);
}

// Original Number: 123456
// Number with Swapped Pairs of Adjacent Digits: 214365
```

201. Capitalization Changer

```rust
fn change_capitalization(input_string: &str) -> String {
    let mut result_string = String::new();

    for char in input_string.chars() {
        // Check if the character is uppercase
        if char.is_uppercase() {
            // Convert uppercase to lowercase
            result_string.push(char.to_lowercase().next().unwrap());
        } else {
            // Convert lowercase to uppercase
            result_string.push(char.to_uppercase().next().unwrap());
        }
    }

    println!("Original String: {}", input_string);
    println!("String with Changed Capitalization: {}", result_string);

    result_string
}

fn main() {
    let input = "Hello World";
    change_capitalization(input);
}

// Original String: Hello World
// String with Changed Capitalization: hELLO wORLD
```

202. Array Halves Swapper

```rust
fn swap_array_halves(arr: &mut Vec<i32>) {
    let length = arr.len();

    // Check if the array has an even length
    if length % 2 != 0 {
        println!("Please provide an array with an even length.");
        return;
    }

    // Calculate the midpoint of the array
    let midpoint = length / 2;

    // Swap the two halves of the array
    for i in 0..midpoint {
        arr.swap(i, midpoint + i);
    }

    println!("Array with Swapped Halves: {:?}", arr);
}

fn main() {
    let mut array = vec![1, 2, 3, 4, 5, 6];
    swap_array_halves(&mut array);
}

// Array with Swapped Halves: [4, 5, 6, 1, 2, 3]
```

203. Sum of Digits in String

```
fn sum_of_digits_in_string(input_string: &str) {
    let mut digit_sum = 0;

    for char in input_string.chars() {
        // Check if the character is a digit
        if char.is_digit(10) {
            // Convert the digit character to its numerical value and add to
sum
            if let Some(digit) = char.to_digit(10) {
                digit_sum += digit;
            }
        }
    }

    println!("Original String: {}", input_string);
    println!("Sum of Digits in the String: {}", digit_sum);
}

fn main() {
    sum_of_digits_in_string("abc123xyz456");
}

// Original String: abc123xyz456
// Sum of Digits in the String: 21
```

204. Sum of Cubes

```
fn sum_of_cubes(up_to_integer: u32) {
    let mut cubes_sum = 0;

    for i in 1..=up_to_integer {
        // Calculate the cube of each integer and add to sum
        cubes_sum += i.pow(3);
    }

    println!("Sum of Cubes from 1 to {}: {}", up_to_integer, cubes_sum);
}

fn main() {
    sum_of_cubes(5);
}

// Sum of Cubes from 1 to 5: 225
```

205. Maximum Integer for Sum

```rust
fn find_max_integer_for_sum(target_sum: u32) {
    let mut current_sum = 0;
    let mut max_integer = 0;

    while current_sum + max_integer + 1 <= target_sum {
        max_integer += 1;
        current_sum += max_integer;
    }

    println!("Maximum Integer (n) for Sum <= {}: {}", target_sum, max_integer);
}

fn main() {
    find_max_integer_for_sum(15);
}

// Maximum Integer (n) for Sum <= 15: 5
```

206. URL Breakdown

```rust
use regex::Regex;
use std::collections::HashMap;

fn break_url(url: &str) {
    let url_regex = Regex::new(r"^(\w+):\/\/([\w.-]+)(\/.*)?$").unwrap();
    let mut url_parts = HashMap::new();

    if let Some(captures) = url_regex.captures(url) {
        url_parts.insert("scheme", captures.get(1).map_or("", |m| m.as_str()));
        url_parts.insert("domain", captures.get(2).map_or("", |m| m.as_str()));
        url_parts.insert("path", captures.get(3).map_or("", |m| m.as_str()));
    } else {
        println!("Invalid URL format.");
        return;
    }

    println!("URL Parts:");
    for (key, value) in &url_parts {
        println!("{}: {}", key, value);
    }
}

fn main() {
    break_url("https://www.example.org/page");
}

// URL Parts:
// path: /page
// scheme: https
// domain: www.example.org
```

207. Sort Strings by Length

```rust
fn sort_strings_by_length(strings_array: &[&str]) {
    let mut sorted_array: Vec<&str> = strings_array.to_vec();
    sorted_array.sort_by_key(|s| s.len());

    println!("Original Array of Strings:");
    println!("{:?}", strings_array);
    println!("Array of Strings Sorted by Length:");
    println!("{:?}", sorted_array);
}

fn main() {
    sort_strings_by_length(&["apple", "banana", "orange", "kiwi", "grape"]);
}

// Original Array of Strings:
// ["apple", "banana", "orange", "kiwi", "grape"]
// Array of Strings Sorted by Length:
// ["kiwi", "apple", "grape", "banana", "orange"]
```