Hochschule RheinMain
Faculty of Design Computer Science Media - Computer Science
Andreas Hellenbrand
Fabio Campos

Rust
Selected Topic in Computer Engineering
LV 7281
Summer 2024

Lab #02

1 rustlings

Take a look at:

- `05_vecs`, `06_move_semantic`, `09_strings`

2 Slice Everything

Implement a function that slices a list into blocks of a specified width N.[1]
Input:

- A list of elements.
- An integer N representing the width of each block.

Output:

- A vector of slices, where each slice represents a block of width N.

Example:

- Input: `[1, 2, 3, 4, 5, 6, 7, 8, 9]`
- N : `3`
- Output: `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

Hints:

- Utilize slices and borrowing to efficiently divide the list into blocks.
- Handle cases where the number of elements is not perfectly divisible by N.

[1] You can use `cargo`, or build your code directly with `rustc`.

```rust
fn main(){
    let my_bites: Vec<u8> = vec![1,2,3,4,5,6,7,8];
    let my_size = 4;

    let my_slices = // your code here
    println!("{:#02X?}", my_slices);
}

fn slicer(/* your code here*/) -> /* your code here*/ {
    // you code here
}
```

3 Change Me

Implement a function that increases each odd number in a list to the next even number.
Input:

- A mutable list of integers.

Output:

- Modify the input list in-place.

Example:

- input: `[1,2,3,4,5,6]`
- output: `[2,2,4,4,6,6]`

Constraints:

- The input list may contain both positive and negative integers.
- Zero is considered an even number.

Hints:

- Use mutable references to modify the input list in-place.
- Determine whether a number is odd or even using the modulo operator %.
- You might need to dereference some values.

```rust
fn main() {
    let mut numbers = vec![1, 2, 3, 4, 5, 6, 7, 8, 9];

    // your code here
    println!("{:#?}", numbers);
}

fn increase_odd_to_next_even(/*your code here*/) {
    // your code here
}
```

You can use `cargo`, or build your code directly with `rustc`.