# Homework for Common Collections

Odd-Harald Lillestø Myhren

July 23, 2023

## Vectors in Rust

### Problem 1: Create a vector in Rust and perform the following operations:

- Insert elements at the end of the vector.
- Remove elements from the vector.
- Update the value of an element in the vector.

### Problem 2: Modify the function below that takes a vector as an argument and return a new vector containing only the unique elements from the input vector.

```
fn main() {
    let input = vec![1, 4, 2, 4, 5, 9, 5, 4, 3, 0, 1, 1];
    let output = get_unique(input);
    assert_eq!(output, vec![1, 4, 2, 5, 9, 3, 0]);
}

fn get_unique(input: Vec<u32>) -> Vec<u32> {
    let mut output = Vec::new();

    // Output should only contain unique values, so you need to use a for-loop
    // on input and check if the current value is in output. If it isn't in the
    // output vector (use input.contains(value)), push it to the output vector.

    return output;
}
```

## HashMap in Rust

### Problem 3: Create a HashMap in Rust and perform the following operations:

- Insert a key-value pair into the HashMap.
- Retrieve a value from the HashMap given its key.
- Update the value associated with a given key.
- Remove a key-value pair from the HashMap given its key.

**Problem 4: Modify the function below to count the frequency of words in a given string using a HashMap. The function should return a HashMap where the keys are words, and the values are the counts of each word.**

```
use std::collections::HashMap;

fn main() {
    let sentence = "This is my amazing phone. This phone is the greatest invention. I can't beli
    let frequency = frequency_of_words(sentence);
    assert_eq!(
        frequency,
        HashMap::from([
            ("I", 2),
            ("This", 2),
            ("amazing", 1),
            ("now.", 1),
            ("can't", 1),
            ("find", 1),
            ("the", 1),
            ("greatest", 1),
            ("believe", 1),
            ("before", 1),
            ("this", 1),
            ("invention.", 1),
            ("phone.", 1),
            ("didn't", 1),
            ("phone", 2),
            ("is", 2),
            ("my", 1)
        ])
    );
}

fn frequency_of_words(input: &str) -> HashMap<&str, usize> {
    let mut output: HashMap<&str, usize> = HashMap::new();

    // First we need to get the words of the input. This can be
    // done with the split_whitespace() iterator:
    for word in input.split_whitespace() {
        // Inside here, we need to check if the word is in the HashMap, by using output.get_mut(
        // If it isn't, we create a new key-value and set output[word] = 1.
        // If the key exists, we need to increment the value for this key
    }

    return output;
}
```