Solution Review: Resizing a Vector

This lesson will give a detailed review of the solution to the challenge in the previous lesson.

We'll cover the following

Solution

```
fn test(my_vec: &mut Vec<u32>)-> &mut Vec<u32>{
   let middle = (my_vec.len())/2;
   my_vec.pop();
   my_vec.remove(middle - 1);
   let mut sum : u32 = 0;
   for v in my_vec.iter()
   {
      sum = sum + v;
   }
   my_vec.push(sum);
   my_vec
}
fn main(){
    let mut v1 = vec![1, 5, 7, 9];
    println!("Original Vector: {:?}", v1);
    println!("Updated Vector: {:?}", test(&mut v1));
    let mut v2 = vec![1, 2, 3, 1, 2, 6];
    println!("Original Vector: {:?}", v2);
    println!("Updated Vector: {:?}", test(&mut v2));
}
```

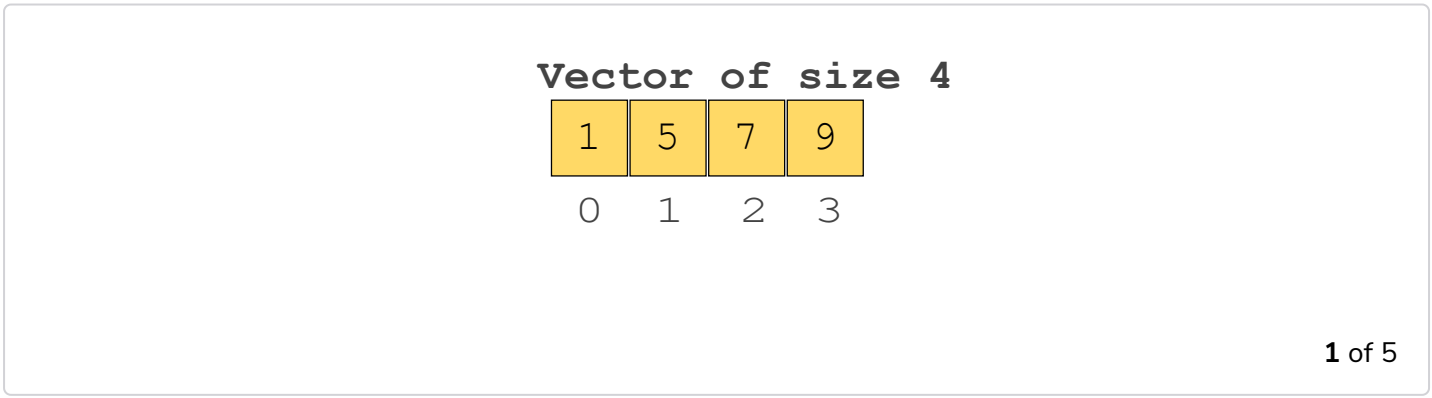
Explanation

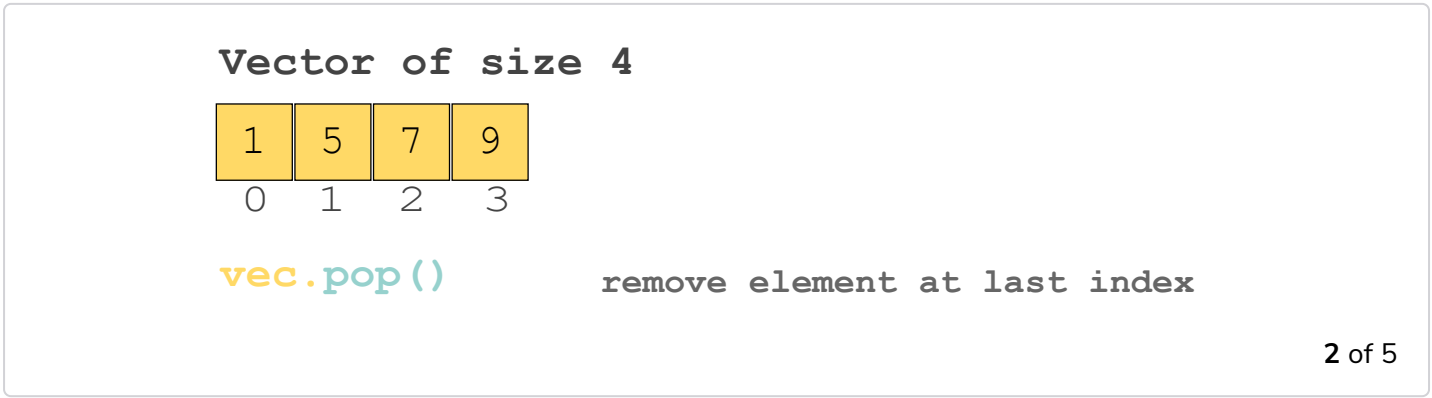
A function `test` is declared with `my_vec` of type `u32` passed to it as a parameter.

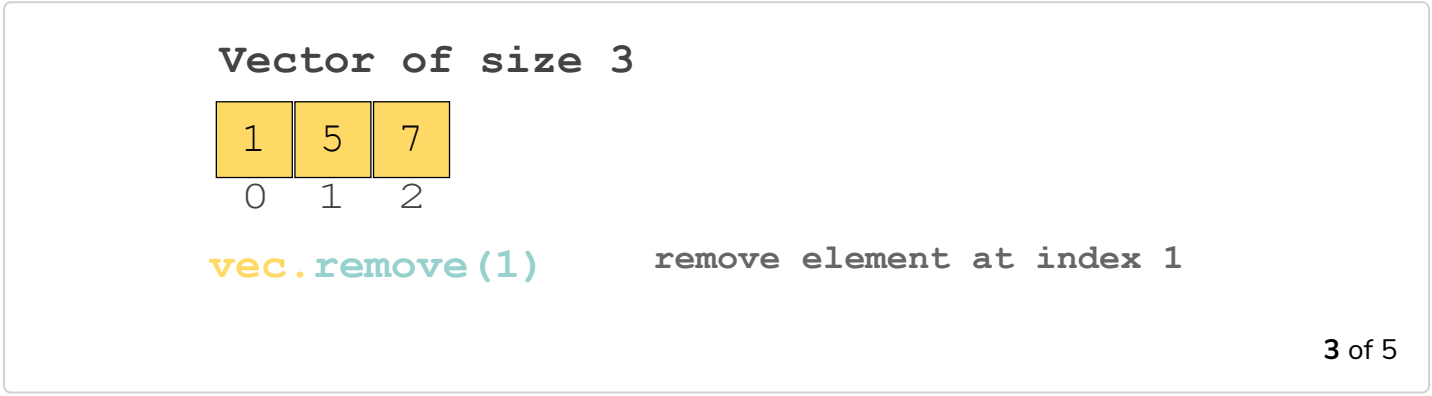
- On **line 2**, a number is removed from the last index using `pop` function, so the number `9` (the last element) gets removed.
- On **line 3**, a number is removed at position 1 using the `remove` function, so the number `5` (the middle element) gets removed.

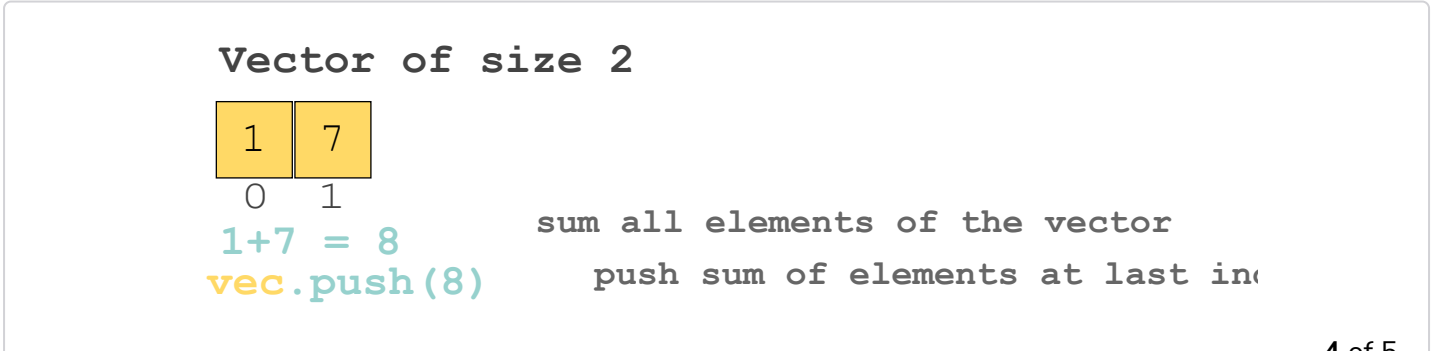
- On **line 4**, a mutable variable `sum` is initialized to 0.
- On **line 5**, a variable `sum` stores the summation of elements of the vector by iterating using a `for` loop over the vector using `my_vec.iter()`
- On **line 6**, a `sum` is added to the last index of `my_vec` using `push` function.
- On **line** 7, `myvec` is returned.

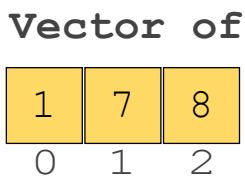
The following illustration explains how the vector resizes according to the above code:











Now that you have learned about vectors, what if you want to group variables having some similarity to be placed under one name in a block of memory, let’s learn about “structs” in the next chapter.