1. User iterator methods to convert a list of strings called ‘strings’ to a list containing the length of each string.

strings.iter().map(|s| s.len()).collect();

2. Select all odd integers from a list of integers called ‘ints’.

ints.iter().filter(|i| i % 2 == 1).collect();

3. Select the length of all strings starting with ‘zy’ from a list called ‘words’.

words.iter().filter(|w| w.starts\_with(‘zy’)).map(|w| w.len())

4. Indicate which variables the following closure will capture and the type of the closure (ie. FnOnce, FnMut, or Fn)

let constant = 42;

let mut list = [0, 1, 2];

let mut index = 0;

let closure = |x| {

list[index] = x;

index += 1;

}

The closure will capture list and index. The closure is an FnMut because it captures and mutates variables from its environment, but it doesn’t move them out and may be called more than once.

5. Implement an iterator method any within the following definition. Any should return true if the closure returns true for any element of the iterator.

pub trait Iterator {

//The type of element the iterator contains

type Item;

/\*

The syntax of the generic function below can be intimidating, so focus on what you

know. Self is an iterator instance (which includes a .next() method), and closure is a

closure which takes an input of type Input and returns a Boolean. The function

overall returns a Boolean.

\*/

fn any<F>(&mut self, closure: F) -> bool where F: FnMut(Self::Item) -> bool {

while let Some(item) = self.next() {

if closure(item) {

true

}

}

false

}

}