1. fn compose(x: i32, f: fn(i32) -> i32, g: fn(i32) -> i32) -> i32 {

f(g(x))

}

1. enum Shape {

Circle(f64),

Rectangle(f64, f64),

}

impl Shape {

fn area(&self) -> f64 {

match \*self {

Shape::Circle(radius) => PI \* radius \* radius,

Shape::Rectangle(length, width) => length \* width,

}

}

}

1. fn f2n<T, F>(f: F, n: u32, x: T) -> T

where

F: Fn(T) -> T,

T: Clone,

{

let mut result = x;

for \_ in 0..n {

result = f(result.clone());

}

result

}