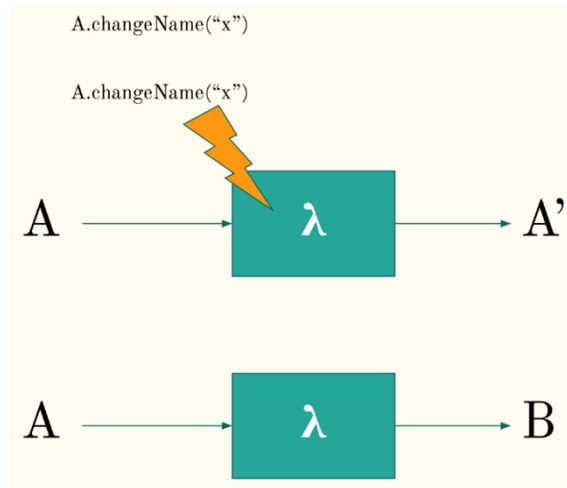


Chapter 1: Introducing Functional Programming



```
life ← {▷1 ω ∨.∧ 3 4 = +/ +/ 1 0 1 ∘.Θ 1 0 1 Φ" <ω}
```

Chapter 2: Treating Functions as First-Class Citizens

No images

Chapter 3: Higher -Order Functions

No images..

Chapter 4: Write Testable Codes with Pure Functions

No images...

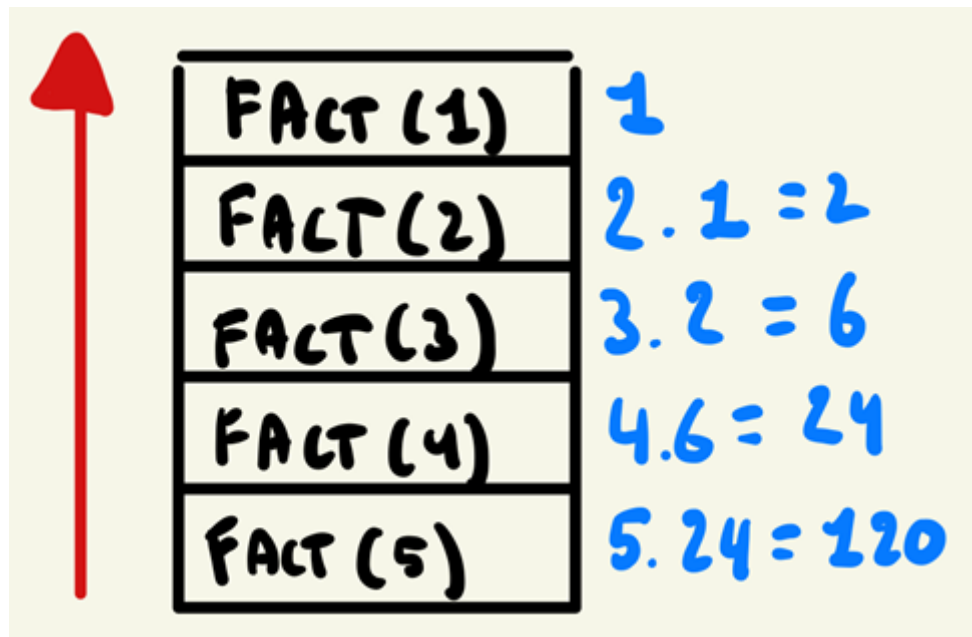
Chapter 5: Immutability

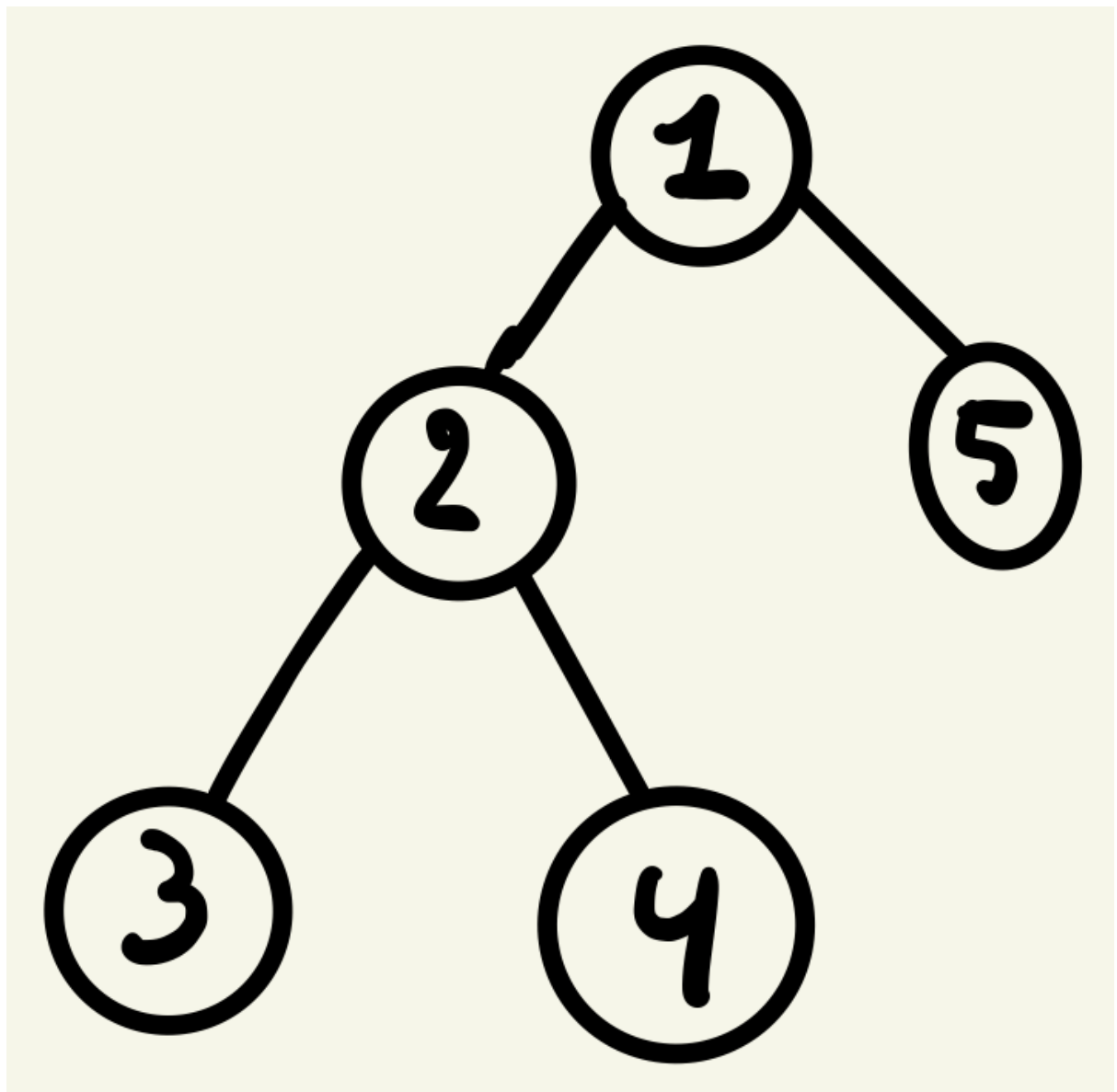
No images...

Chapter 6: Three Common Categories of Functions

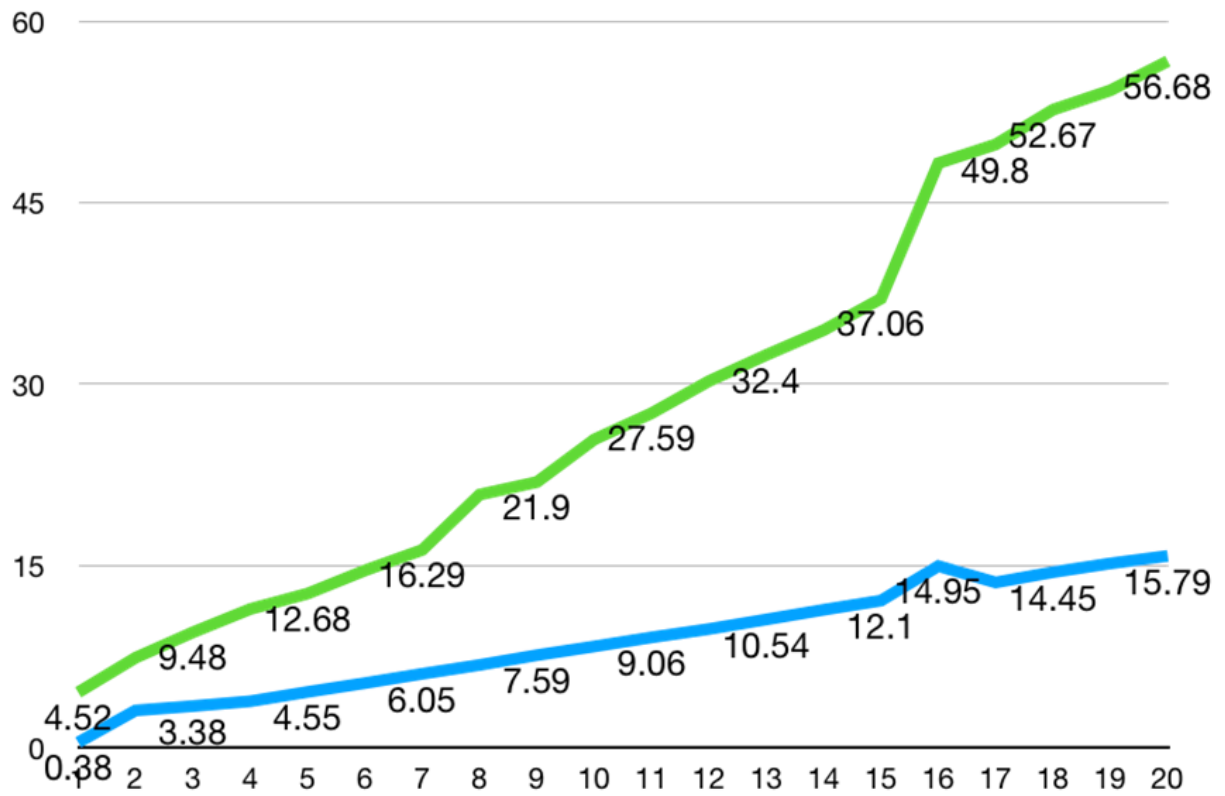
No images...

Chapter 7: Recursion






Iterative vs Recursive factorial in ns/op

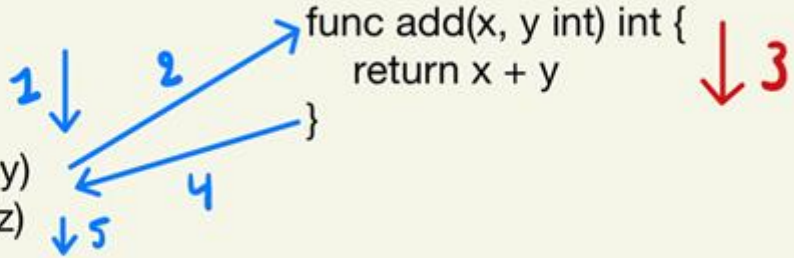


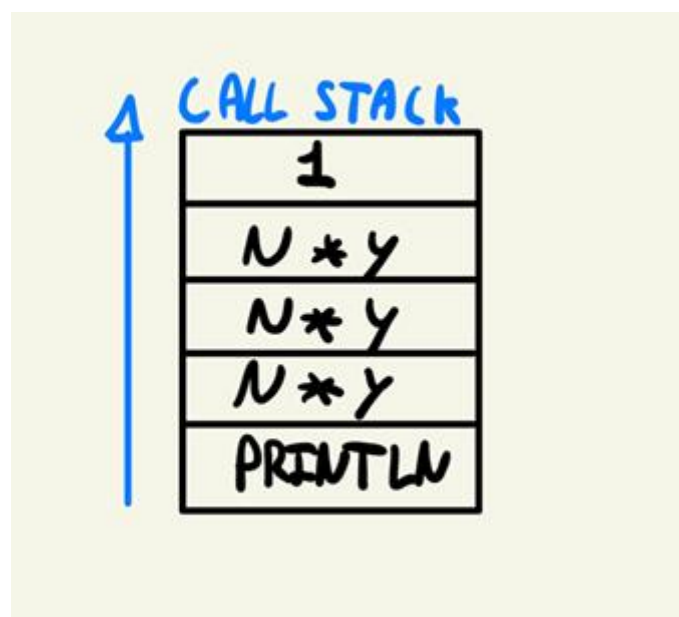
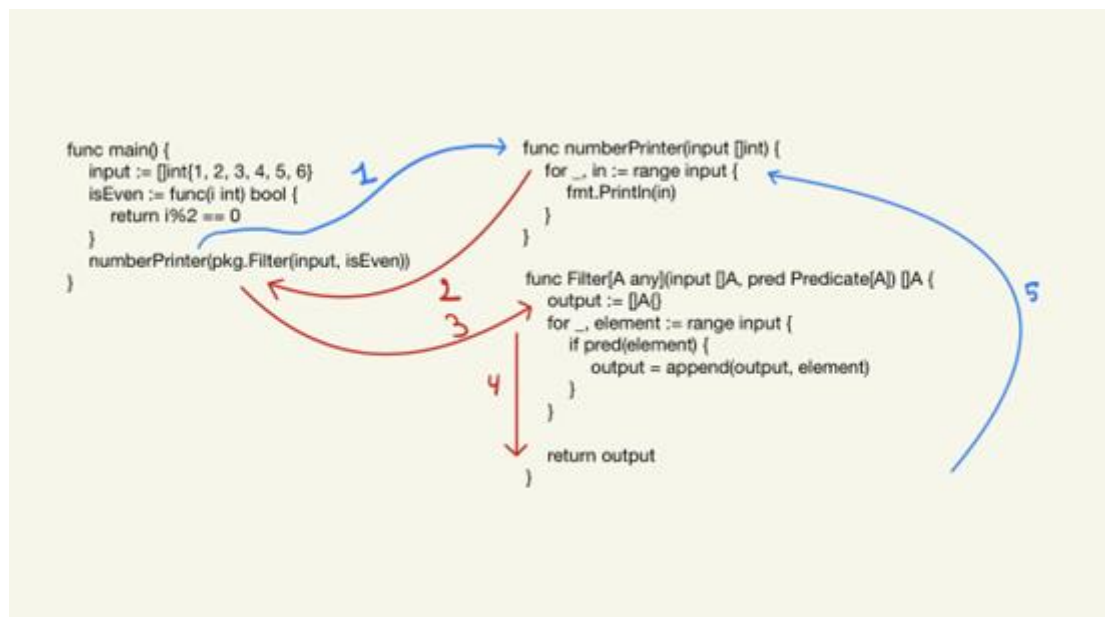
Chapter 8: Readable Function Composition with Fluent Programming

```
func main() {  
    x := 3  
    y := 4  
    z := x + y  
    fmt.Println(z)  
}
```

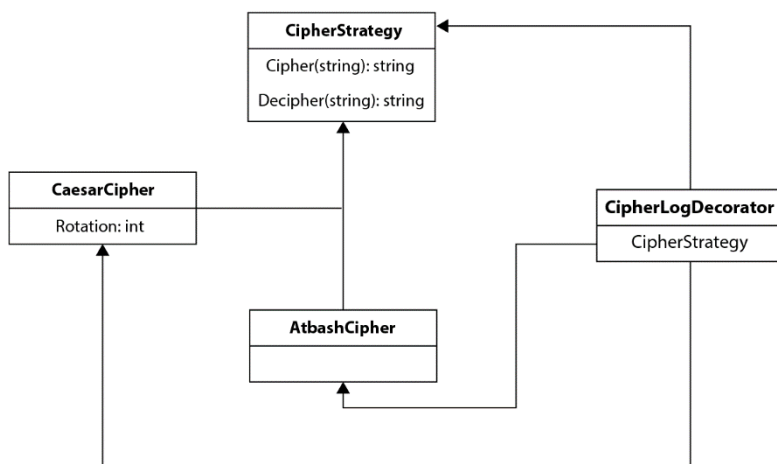
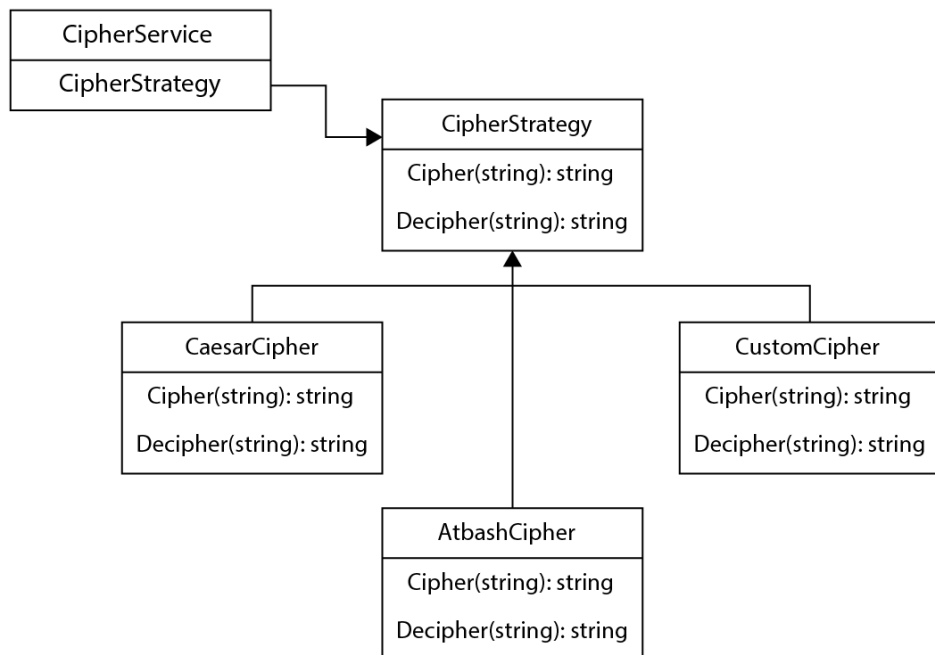


```
func main() {  
    x := 3  
    y := 4  
    z := add(x, y)  
    fmt.Println(z)  
}  
  
func add(x, y int) int {  
    return x + y  
}
```





Chapter 9: Functional Design Patterns

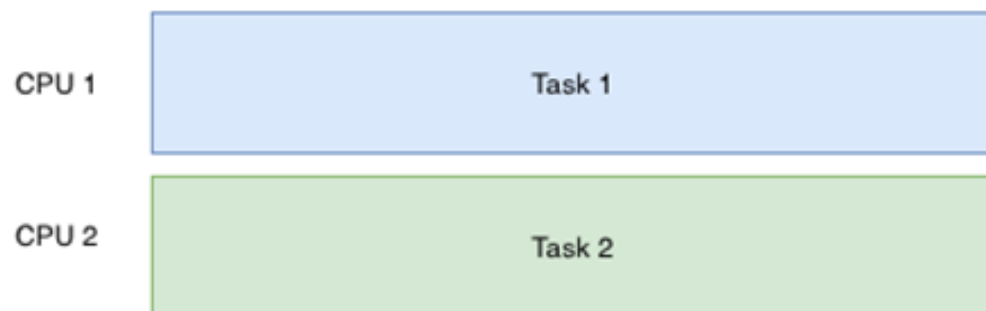


Chapter 10: Thinking functionally to solve problems

Concurrency



Parallelism



Chapter 11: Functional Programming Libraries

No images...