Basic concepts

1. Coding

   A. Give an example of code smell (1) that usually happened in OOP. Explain and give the preventive action

   Empty/Temporary Field

   Temporary field is smells that occurs when a field that required by objects but the value only appear only under certain circumstances, outside of that circumstances the value is empty.

   Oftentimes, temporary fields are created for use in an algorithm that requires a large amount of inputs. So instead of creating a large number of parameters in the method, the programmer decides to create fields for this data in the class. These fields are used only in the algorithm and go unused the rest of the time.

   This kind of code is tough to understand. You expect to see data in object fields but for some reason they're almost always empty.

   Solution

   Temporary fields and all code operating on them can be put in a separate class via Extract Class. In other words, you're creating a method object, achieving the same result as if you would perform Replace Method with Method Object.

   Introduce Null Object and integrate it in place of the conditional code which was used to check the temporary field values for existence.

   B. Explain briefly about Dependency Injection, and why is it important ?

   What is Dependency Injection :

   Dependency Injection (DI) is a technique or a concept in software development that allows a service to be used/injected in a way that is completely independent of any client consumption, or simply when an object is needed by another object without using inheritance concept.

   Why is it important?

   Dependency Injection is important because it makes the code (object) independent, so increase the reusability, and easier to refactor, also easy to test.

2. Rest API
   A. Give 1 example each, of what do’s and don’ts when you handle request while using these method :

- POST

  a. Do use POST whenever you have to do something that feels RPC-like.
  b. Don't use POST to a request where only retrieve information without sending any information

- GET

  a. Do use GET whenever you make a request just to receive information.
  b. Don’t use GET when you make a request that change the database state, like creating, updating or deleting data.

Basic coding

Imagine you're working in a company and you're told to make a design system and transaction flow. How would you make the most suitable for following users and case:

User Side

- User register
- User input the address
- User choose the products to be purchased with subscription and/or one-time purchase scheme
- User pay the bill
- User skip the delivery due to certain reasons (ex: They have other agenda that prevent them to receive the delivered goods)
- User cancel the order

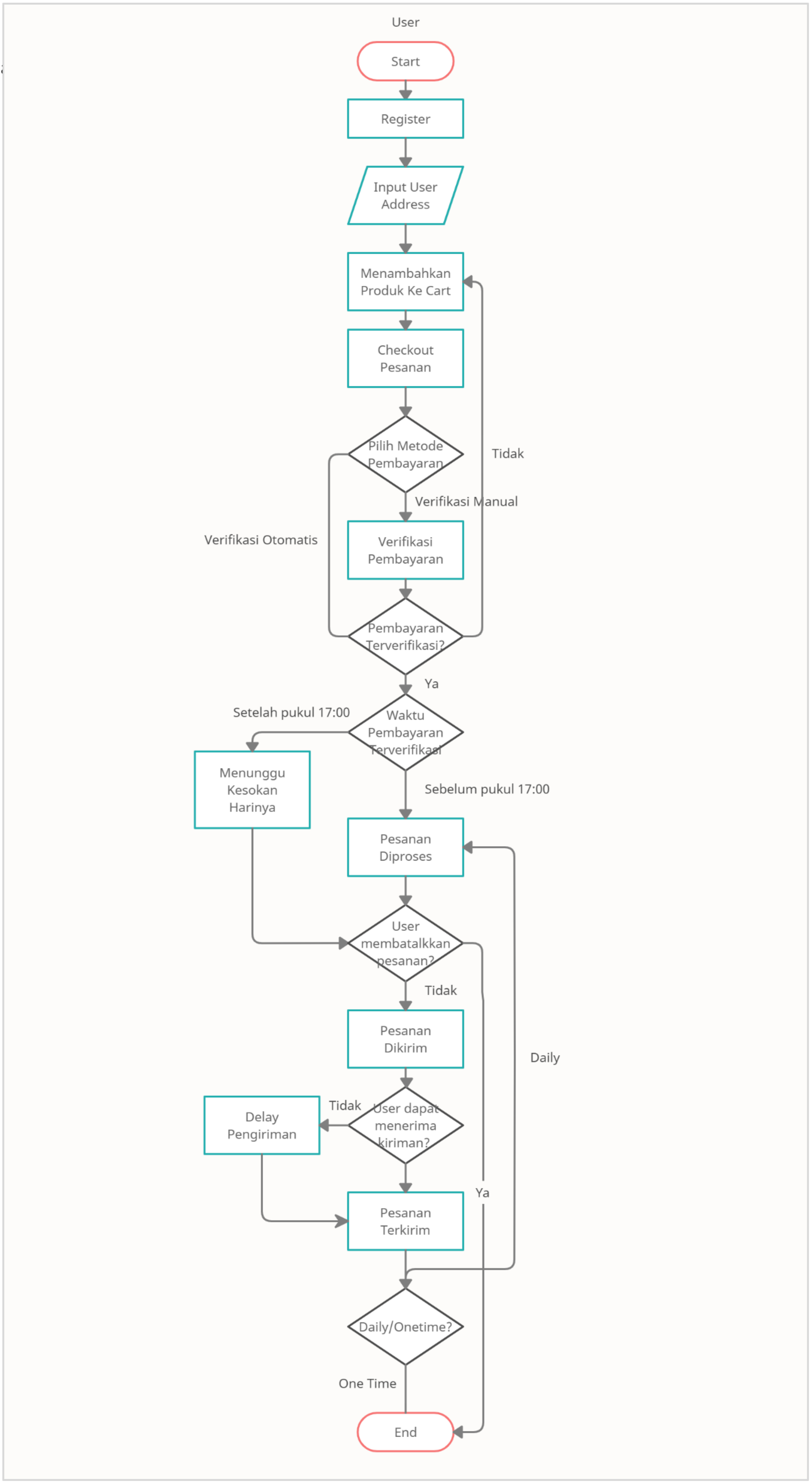
Supplier Side

- Supplier register as seller
- Supplier create the store and complete the address
- Supplier create products that can be purchased either daily or one-time purchase
- Supplier determine the price of each product
- Supplier determine the selling area

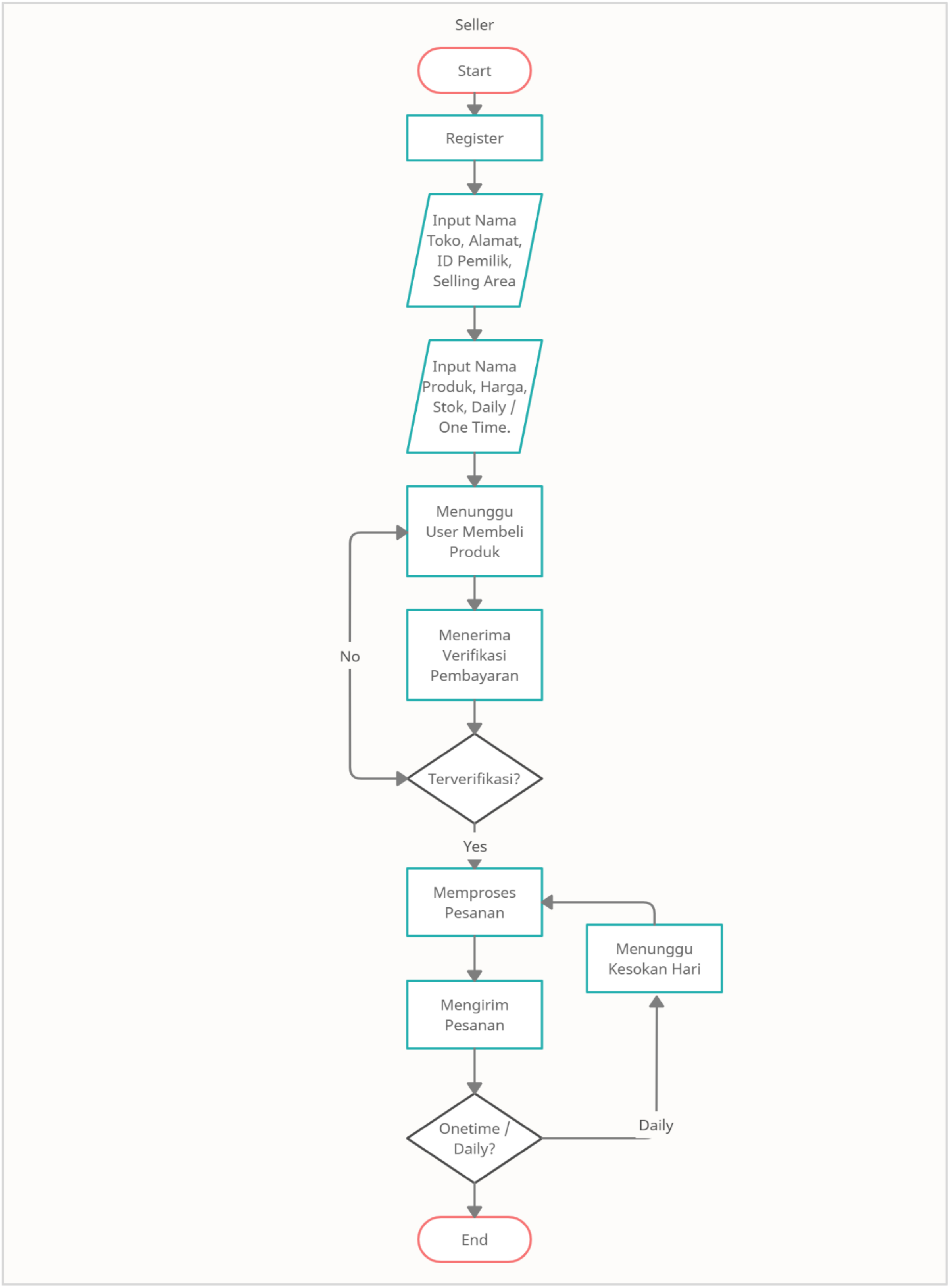
Additional:

- If a product can be sold by more than one seller, define the correct algorithm to determine which order to be sent from which seller (assuming the closest mileage and route)!
- There is a cut-off time everyday, which is the latest time an order can be placed for the next day delivery. All orders placed beyond cut-off time will automatically delivered on the day after tomorrow.

Seig


User
Start
Register
Input User Address
Menambahkan Produk Ke Cart
Checkout Pesanan
Pilih Metode Pembayaran
Tidak
Verifikasi Manual
Verifikasi Otomatis
Verifikasi Pembayaran
Pembayaran Terverifikasi?
Ya
Setelah pukul 17:00
Waktu Pembayaran Terverifikasi
Menunggu Kesokan Harinya
Sebelum pukul 17:00
Pesanan Diproses
User membatalkkan pesanan?
Tidak
Pesanan Dikirim
Daily
Delay Pengiriman
Tidak
User dapat menerima kiriman?
Ya
Pesanan Terkirim
Daily/Onetime?
One Time
End

Seiga AL Ghifari – Backend Prelim Test


Seller
Start
Register
Input Nama Toko, Alamat, ID Pemilik, Selling Area
Input Nama Produk, Harga, Stok, Daily / One Time.
Menunggu User Membeli Produk
Menerima Verifikasi Pembayaran
Terverifikasi?
No
Yes
Memproses Pesanan
Menunggu Kesokan Hari
Mengirim Pesanan
Onetime / Daily?
Daily
End

Algorithm

1. Nick works at a clothing store. He has a large pile of socks that he must pair by color for sale. Given an array of integers representing the color of each sock, determine how many pairs of socks with matching colors there are.

   For example, there are n= 7 socks with colors ar = [1, 2, 1, 2, 1, 3, 2]. There is one pair of color 1 and one of color 2. There are three odd socks left, one of each color. The number of pairs is 2.

   Function Description

   Complete the sock merchant function in the editor below. It must return an integer representing the number of matching pairs of socks that are available.

   sockMerchant has the following parameter(s):

   - n: the number of socks in the pile
   - ar : the colors of each sock

   Input Format

   The first line contains an integer n, the number of socks represented inar .

   The second line contains n space-separated integers describing the colors ar [i] of the socks in the pile. Constraints

   $1 \leq n \leq 100$

   $1 \leq ar[i] \leq 100 \ where \ 0 \leq i < n$

   Output Format

   Return the total number of matching pairs of socks that Nick can sell. **Sample**

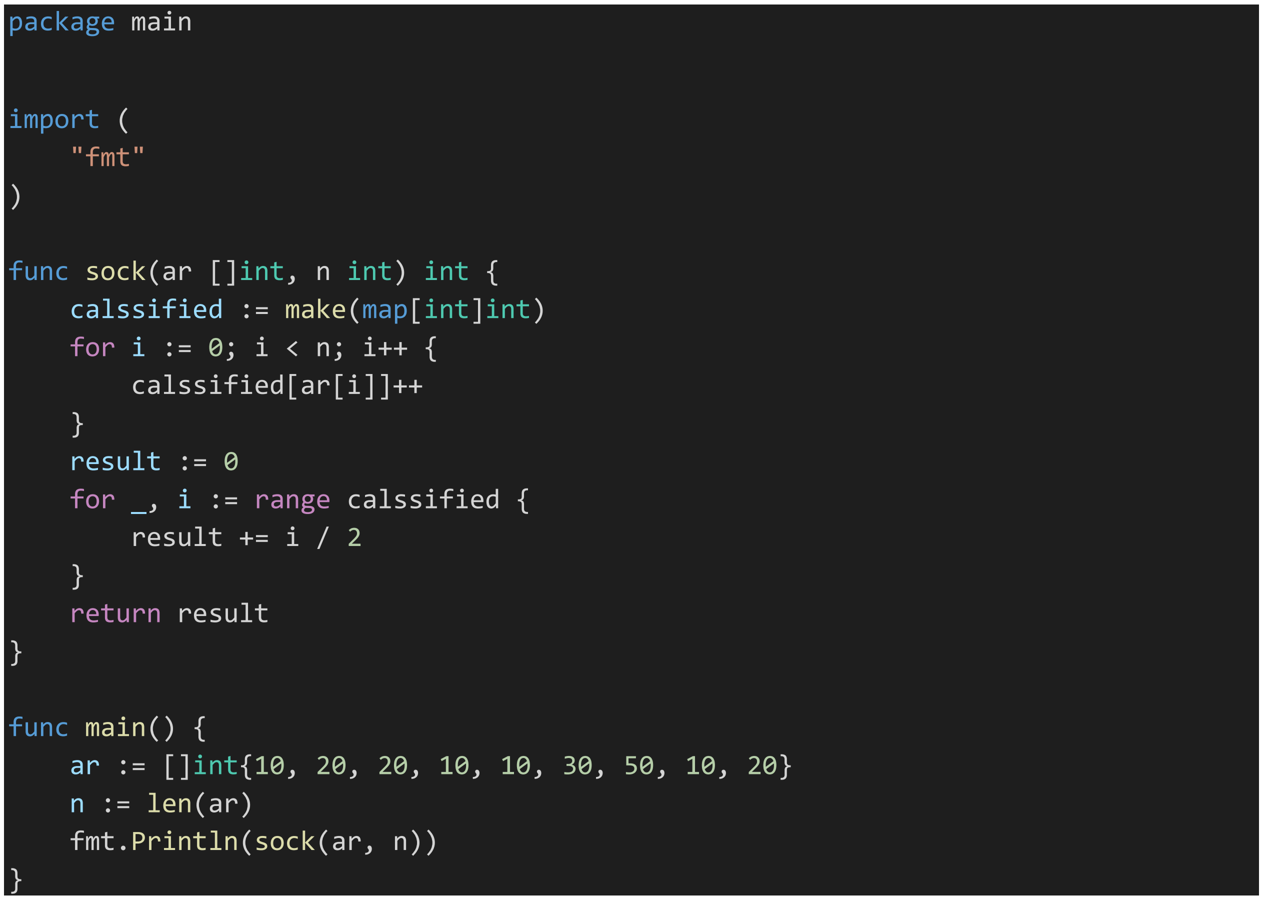
   Input

```
9
10 20 20 10 10 30 50 10 20
```

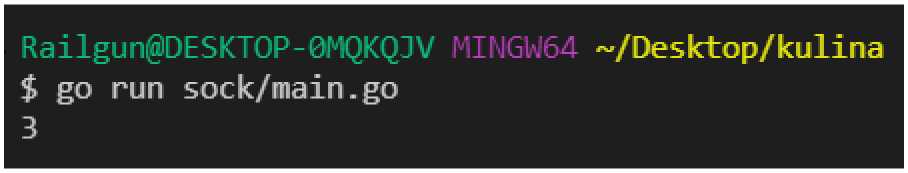
   Sample output

```
3
```

Seiga AL Ghifari – Backend Prelim Test



```go
package main


import (
    "fmt"
)

func sock(ar []int, n int) int {
    calssified := make(map[int]int)
    for i := 0; i < n; i++ {
        calssified[ar[i]]++
    }
    result := 0
    for _, i := range calssified {
        result += i / 2
    }
    return result
}

func main() {
    ar := []int{10, 20, 20, 10, 10, 30, 50, 10, 20}
    n := len(ar)
    fmt.Println(sock(ar, n))
}
```



```
Railgun@DESKTOP-0MQKQJV MINGW64 ~/Desktop/kulina
$ go run sock/main.go
3
```

2. Bill is an avid hiker. He tracks his hikes meticulously, paying close attention to small details like topography. During his last hike he took exactly n steps. For every step he took, he noted if it was an uphill, U , or a downhill, D step. Gary's hikes start and end at sea level and each step up or down represents a 1 unit change in altitude. We define the following terms:
    - A mountain is a sequence of consecutive steps above sea level, starting with a step up from sea level and ending with a step down to sea level.
    - A valley is a sequence of consecutive steps below sea level, starting with a step down from sea level and ending with a step up to sea level.

   Given Gary's sequence of up and down steps during his last hike, find and print the number of valleys he walked through.

   For example, if Gary's path is $s = [D\ D\ U\ U\ U\ U\ D\ D]$, he first enters a valley 2 units deep. Then he climbs out an up onto a mountain 2 units high. Finally, he returns to sea level and ends his hike.

   Function Description

   Complete the countingValleys function in the editor below. It must return an integer that denotes the number of valleys Gary traversed.

   countingValleys has the following parameter(s):

    - n: the number of steps Gary takes
    - s: a string describing his path

   Input Format

   The first line contains an integer n , the number of steps in Gary's hike.

   The second line contains a single string s , of $n$characters that describe his path. **Constraints**

    - $2 \leq n \leq 10^6$
    - $s[i] \in \{ U\ D \}$

   Output Format

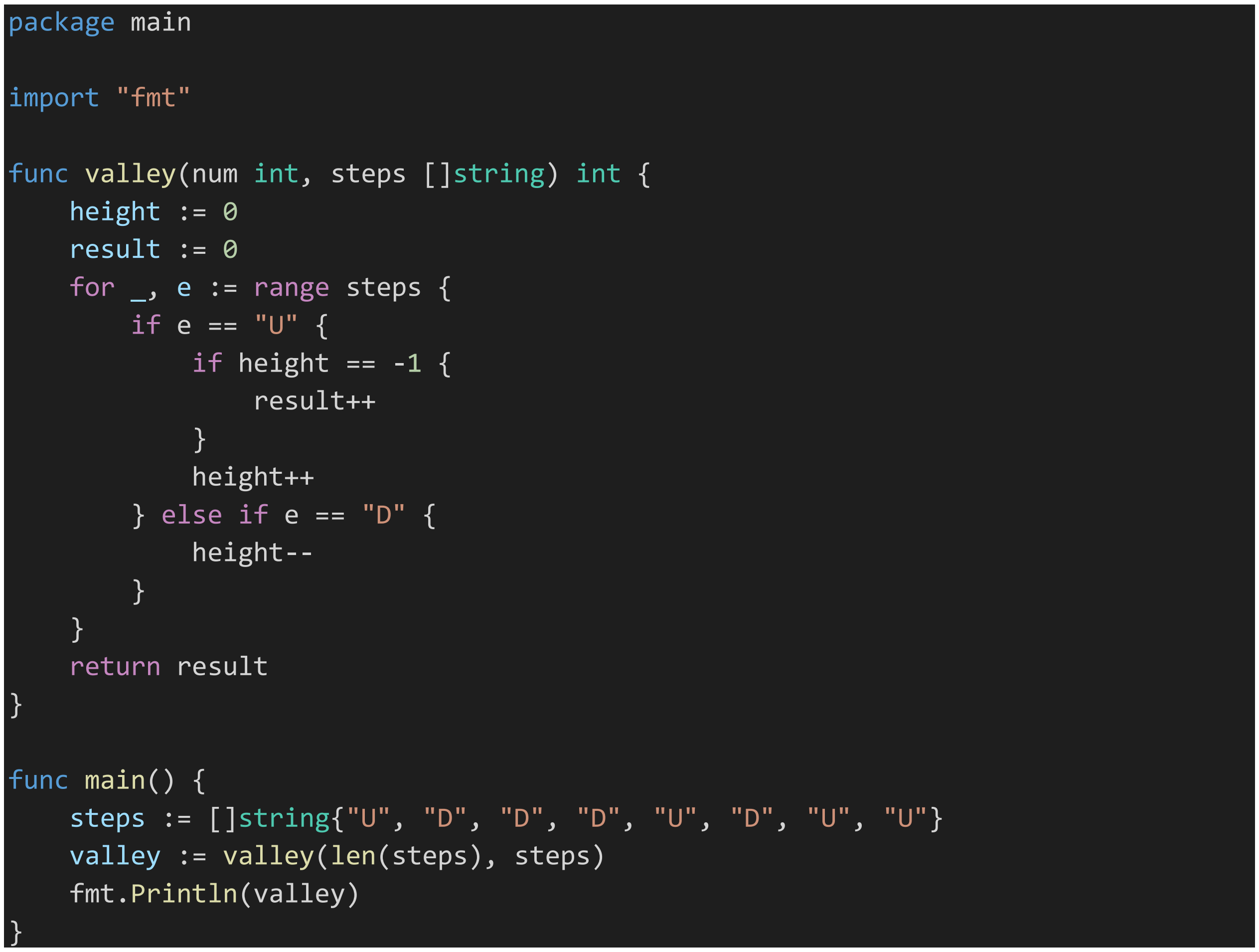
   Print a single integer that denotes the number of valleys Gary walked through during his hike.

   Sample Input

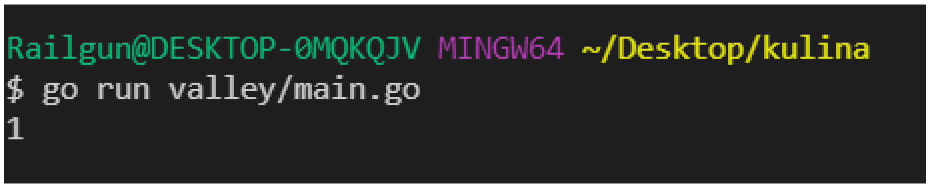
   ```
   8
   U D D D U D U U
   ```

   Sample Output

   ```
   1
   ```



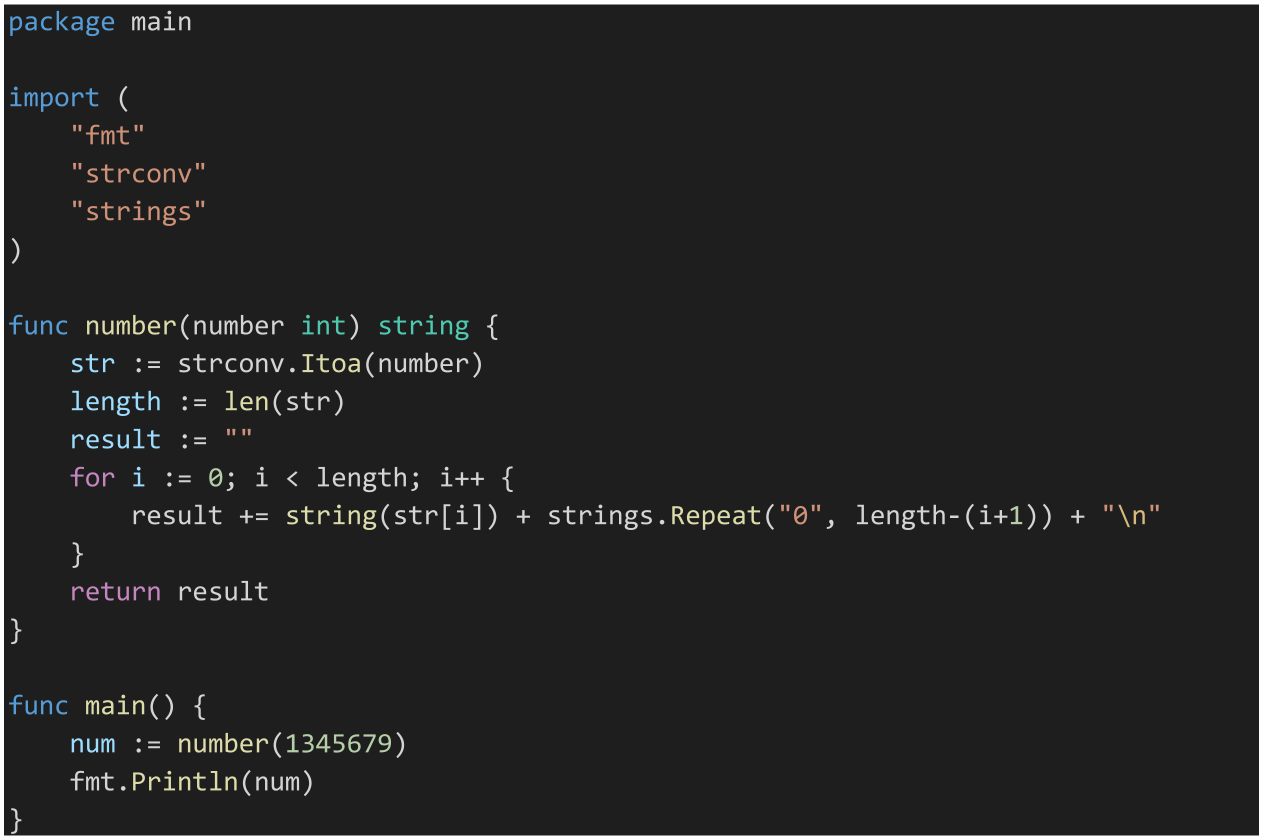
```go
package main

import "fmt"

func valley(num int, steps []string) int {
    height := 0
    result := 0
    for _, e := range steps {
        if e == "U" {
            if height == -1 {
                result++
            }
            height++
        } else if e == "D" {
            height--
        }
    }
    return result
}

func main() {
    steps := []string{"U", "D", "D", "D", "U", "D", "U", "U"}
    valley := valley(len(steps), steps)
    fmt.Println(valley)
}
```



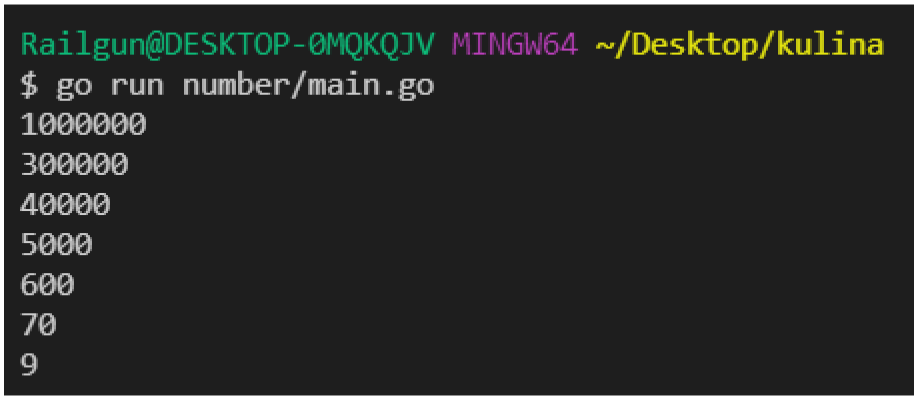
```
Railgun@DESKTOP-0MQKQJV MINGW64 ~/Desktop/kulina
$ go run valley/main.go
1
```

3. There is an input number: 1.345.679
   Write pseudo code (use **GoLang**) that produces following output:

   1000000
   300000
   40000
   5000
   600
   70
   9



```go
package main

import (
    "fmt"
    "strconv"
    "strings"
)

func number(number int) string {
    str := strconv.Itoa(number)
    length := len(str)
    result := ""
    for i := 0; i < length; i++ {
        result += string(str[i]) + strings.Repeat("0", length-(i+1)) + "\n"
    }
    return result
}

func main() {
    num := number(1345679)
    fmt.Println(num)
}
```

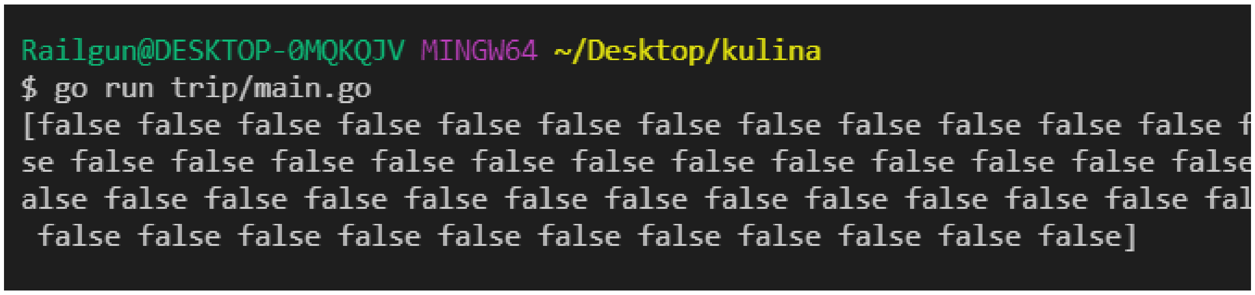


```
Railgun@DESKTOP-0MQKQJV MINGW64 ~/Desktop/kulina
$ go run number/main.go
1000000
300000
40000
5000
600
70
9
```

4. Andrew walks through 100 switches from point A to point B with 1 to 100 as the number. At the first trip, Andrew presses all of the switches so lamps are on. Second trip, Andrew only presses switches that multiplying of two. The next trip, Andrew presses switches that multiplying of three. Andrew repeats his trips from A to B for 100 times. Write down the code to determine how many lamps will turn on after 100 trips from A to B.

```go
package main

import "fmt"

func trip(ar []bool, trip int) []bool {
    newArr := ar
    for i := 0; i < trip; i++ {
        for i := range newArr {
            newArr[i] = !newArr[i]
        }
        for i := range newArr {
            if (i+1)%2 == 0 {
                newArr[i] = !newArr[i]
            }
        }
        for i := range newArr {
            if (i+1)%3 == 0 {
                newArr[i] = !newArr[i]
            }
        }
    }
    return newArr
}

func lamp(sum int) []bool {
    ar := make([]bool, 100)
    for i := 0; i < 100; i++ {
        ar[i] = false
    }
    return ar
}

func main() {
    lamp := lamp(100)
    result := trip(lamp, 100)
    fmt.Println(result)
}
```

Seiga AL Ghifari – Backend Prelim Test



```
Railgun@DESKTOP-0MQKQJV MINGW64 ~/Desktop/kulina
$ go run trip/main.go
[false false false false false false false false false false false false f
se false false false false false false false false false false false false
alse false false false false false false false false false false false fal
 false false false false false false false false false false false]
```