Netcentric lab 1

Nguyen Manh Viet Khoi
ITCSIU21081

1.
distance.go

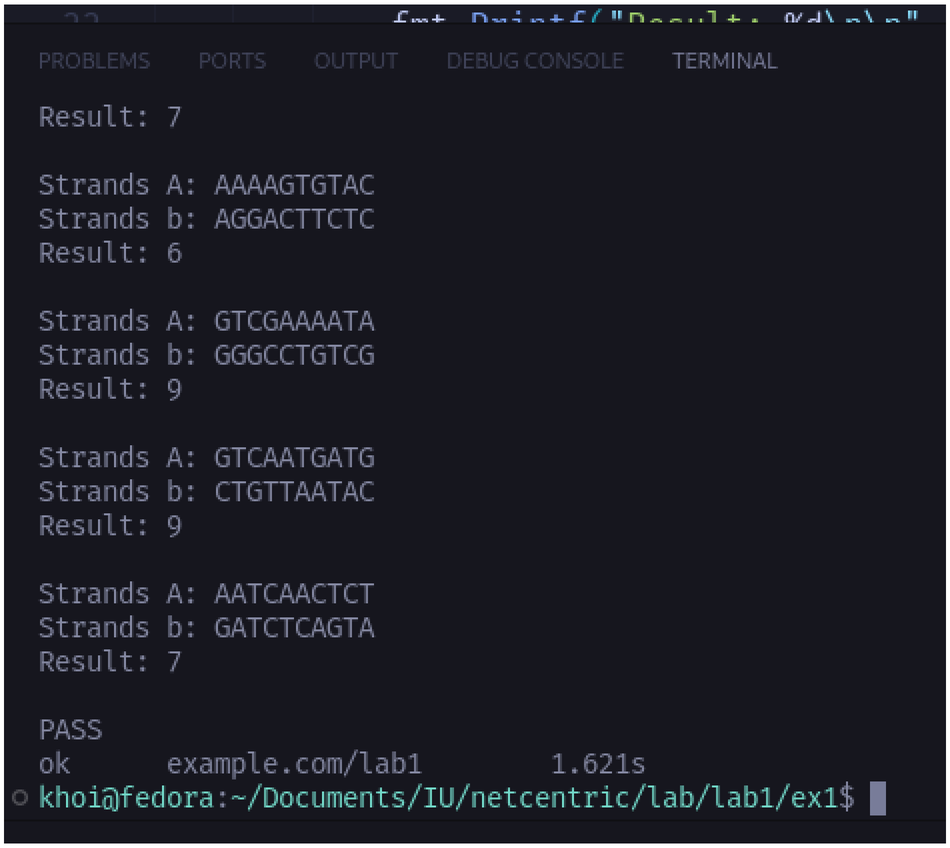
```go
package main

import (
    "errors"
    "math/rand"
)

func Distance(a, b []byte) (int, error) {
    if len(a) != len(b) {
        return 0, errors.New("STRANDS SHOULD HAVE SAME LENGTH")
    }

    distance := 0;
    for i := 0; i < len(a); i++ {
        if a[i] != b[i] {
            distance++
        }
    }

    return distance, nil;
}

func GenerateDNA(length int) ([]byte, []byte) {
    const dna = "ATGC"
    a := make([]byte, length)
    b := make([]byte, length)
    for i := range a {
        a[i] = dna[rand.Intn(len(dna))]
        b[i] = dna[rand.Intn(len(dna))]
```

```go
   }
   return a, b
}
```

Distance_test.go

```go
package main

import (
   "fmt"
   "testing"
)

func TestDistance(t *testing.T) {
   length := 10
   var a []byte
   var b []byte

   for i:= 0; i < 10000; i++ {
       a, b = GenerateDNA(length)
       result, err := Distance(a, b)
       if err != nil {
           t.Errorf("Got error")
       } else {
           t.Log(result)
           fmt.Printf("Strands A: %s\n", a)
           fmt.Printf("Strands b: %s\n", b)
           fmt.Printf("Result: %d\n\n", result)
       }
   }
}
```

Result



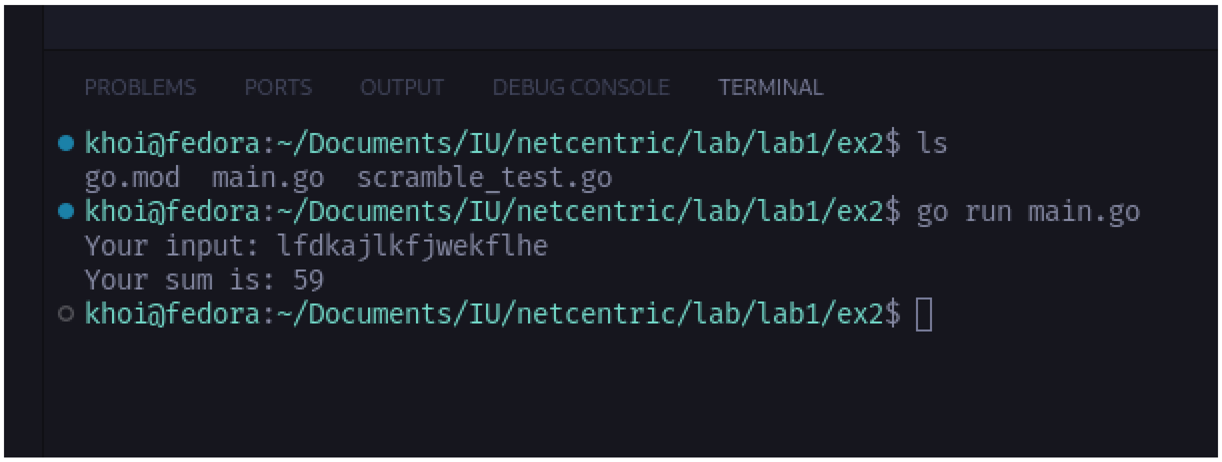
```
Result: 7

Strands A: AAAAGTGTAC
Strands b: AGGACTTCTC
Result: 6

Strands A: GTCGAAAATA
Strands b: GGGCCTGTCG
Result: 9

Strands A: GTCAATGATG
Strands b: CTGTTAATAC
Result: 9

Strands A: AATCAACTCT
Strands b: GATCTCAGTA
Result: 7

PASS
ok      example.com/lab1        1.621s
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex1$
```

- In this exercise I make a test file for this program. Therefore, if you want to run just type go test in the terminal.

2.
main.go

```go
package main

import "fmt"

func main() {

   a := "lfdkajlkfjwekflhe"
   sum := 0
   fmt.Printf("Your input: %s\n", a);
   for _, c := range a {
       sum += score(string(c))
```

```
    }
    fmt.Printf("Your sum is: %d\n", sum)
}

func score(letter string) (int) {
    switch letter {
        case "d", "g" : return 2
        case "b", "c", "m", "p" : return 3
        case "f", "h", "v", "w", "y": return 4
        case "k" : return 5
        case "j", "x" : return 8
        case "q", "z" : return 10
    default: return 1
    }
}
```



```
● khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex2$ ls
  go.mod  main.go  scramble_test.go
● khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex2$ go run main.go
  Your input: lfdkajlkfjwekflhe
  Your sum is: 59
○ khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex2$ 
```

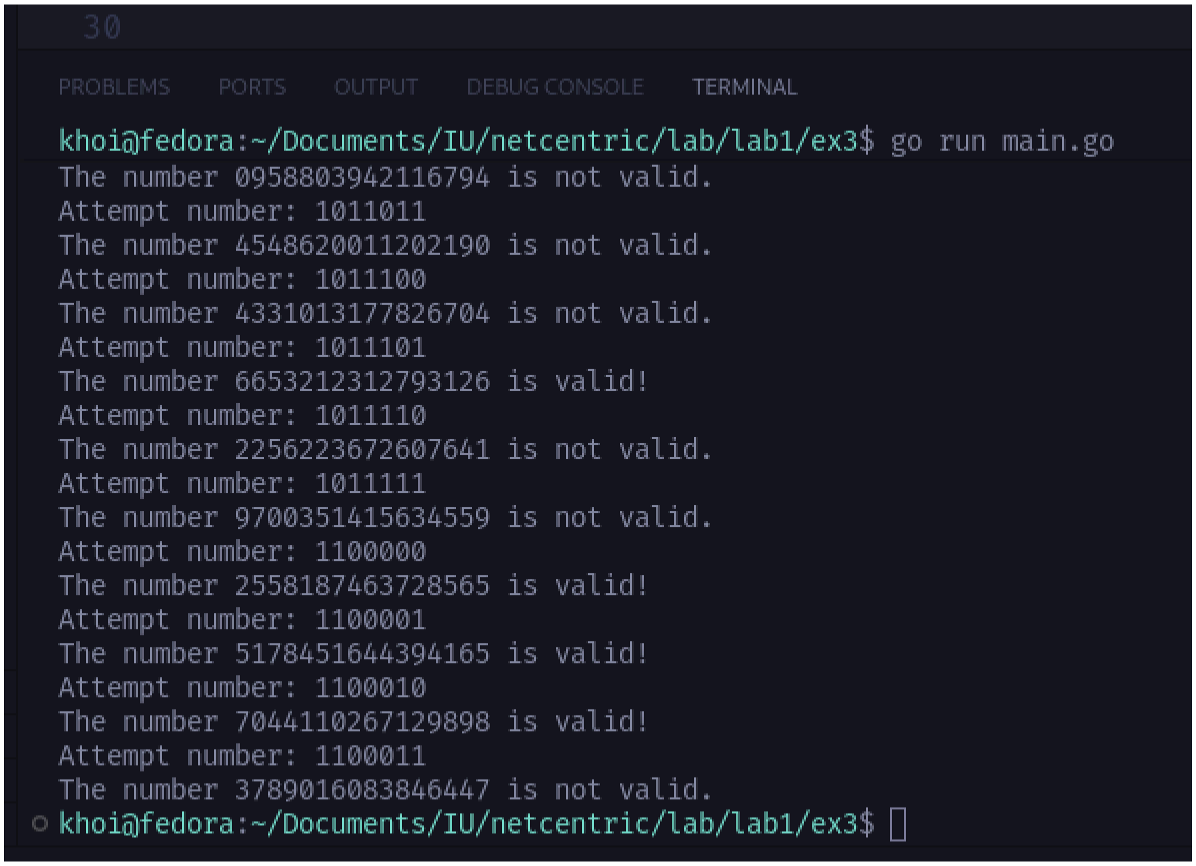
- The program loops through the string and uses a switch case to determine the value of the character.

3.

```
package main

import (
    "fmt"
    "math/rand"
```

```go
    "strconv"
    "strings"
)

func isValidNumber(input string) bool {
    input = strings.ReplaceAll(input, " ", "")

    sum := 0
    doubleNext := false

    for i := len(input) - 1; i >= 0; i-- {
        digit, _ := strconv.Atoi(string(input[i]))

        if doubleNext {
            digit *= 2
            if digit > 9 {
                digit -= 9
            }
        }
        sum += digit
        doubleNext = !doubleNext
    }
    return sum%10 == 0
}

func GenerateNumber(length int) (string) {
    const letters = "0123456789"
    a := make([]byte, length)
    for i := range a {
        a[i] = letters[rand.Intn(len(letters))]
    }
    return string(a)
}
```

```go
func main () {
   for i := 0; i < 100; i++ {
       input := GenerateNumber(16)
       fmt.Printf("Attempt number: %b\n", i)
       if isValidNumber(input) {
           fmt.Printf("The number %s is valid!\n", input)
       } else {
           fmt.Printf("The number %s is not valid.\n", input)
       }
   }
}
```



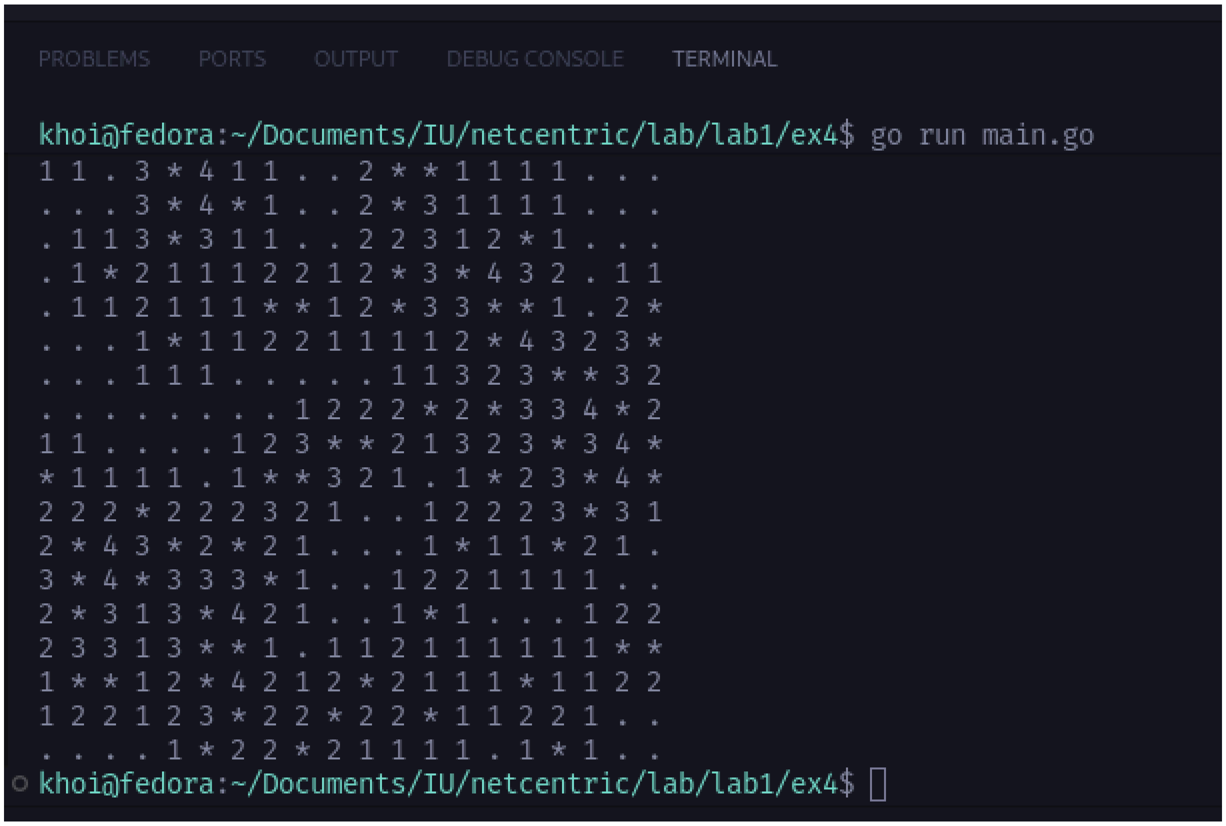
```
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex3$ go run main.go
The number 0958803942116794 is not valid.
Attempt number: 1011011
The number 4548620011202190 is not valid.
Attempt number: 1011100
The number 4331013177826704 is not valid.
Attempt number: 1011101
The number 6653212312793126 is valid!
Attempt number: 1011110
The number 2256223672607641 is not valid.
Attempt number: 1011111
The number 9700351415634559 is not valid.
Attempt number: 1100000
The number 2558187463728565 is valid!
Attempt number: 1100001
The number 5178451644394165 is valid!
Attempt number: 1100010
The number 7044110267129898 is valid!
Attempt number: 1100011
The number 3789016083846447 is not valid.
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex3$ 
```

4.
main.go

```go
package main

import (
   "fmt"
   "math/rand"
)

func createField(width int, height int, mines int) [][]string {
   field := make([][]string, height+2)
   // an array that contains the coordinates of the mines
   minesCoordinates := make([][]int, mines)
   // init field
   for i := 0; i < height+2; i++ {
       field[i] = make([]string, width+2)
       for j := 0; j < width+2; j++ {
           field[i][j] = "."
       }
   }
   // init minesCoordinates
   for i := 0; i < mines; i++ {
       minesCoordinates[i] = make([]int, 2)
   }
   // fill field with mines
   for i := 0; i < mines; i++ {
       for {
           x := rand.Intn(width) + 1
           y := rand.Intn(height) + 1
           place := false
           // check if there is already a mine at this position
           if minesCoordinates[x][0] == x &&
minesCoordinates[y][1] == y {
               continue
           }
           // place mine
           minesCoordinates[i][0] = x
           minesCoordinates[i][1] = y
           field[y][x] = "*"

           place = true
           if place {
               break
           }
       }

   }
   // fill field with numbers
   for i := 1; i < height+1; i++ {
       for j := 1; j < width+1; j++ {
           if field[i][j] == "." {
               count := 0
               for k := i - 1; k <= i+1; k++ {
                   for l := j - 1; l <= j+1; l++ {
                       if field[k][l] == "*" {
                           count++
                       }
                   }
               }
               if count > 0 {
                   field[i][j] = fmt.Sprintf("%d", count)
               }
           }
       }
   }
   return field
}
func printField(field [][]string) {
   for i := 1; i < len(field)-1; i++ {
       for j := 1; j < len(field[i])-1; j++ {
           fmt.Printf("%s ", field[i][j])
       }
```

```go
        fmt.Println()
    }
}

func main() {
    board := createField(20, 25, 99)
    printField(board)
    // fmt.Println(board)
}
```

Result



```
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex4$ go run main.go
1 1 . 3 * 4 1 1 . . 2 * * 1 1 1 1 . . .
. . . 3 * 4 * 1 . . 2 * 3 1 1 1 1 . . .
. 1 1 3 * 3 1 1 . . 2 2 3 1 2 * 1 . . .
. 1 * 2 1 1 1 2 2 1 2 * 3 * 4 3 2 . 1 1
. 1 1 2 1 1 1 * * 1 2 * 3 3 * * 1 . 2 *
. . . 1 * 1 1 2 2 1 1 1 1 2 * 4 3 2 3 *
. . . 1 1 1 . . . . . 1 1 3 2 3 * * 3 2
. . . . . . . . 1 2 2 2 * 2 * 3 3 4 * 2
1 1 . . . . 1 2 3 * * 2 1 3 2 3 * 3 4 *
* 1 1 1 1 . 1 * * 3 2 1 . 1 * 2 3 * 4 *
2 2 2 * 2 2 2 3 2 1 . . 1 2 2 2 3 * 3 1
2 * 4 3 * 2 * 2 1 . . . 1 * 1 1 * 2 1 .
3 * 4 * 3 3 3 * 1 . . 1 2 2 1 1 1 1 . .
2 * 3 1 3 * 4 2 1 . . 1 * 1 . . . 1 2 2
2 3 3 1 3 * * 1 . 1 1 2 1 1 1 1 1 1 * *
1 * * 1 2 * 4 2 1 2 * 2 1 1 1 * 1 1 2 2
1 2 2 1 2 3 * 2 2 * 2 2 * 1 1 2 2 1 . .
. . . . 1 * 2 2 * 2 1 1 1 1 . 1 * 1 . .
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex4$
```

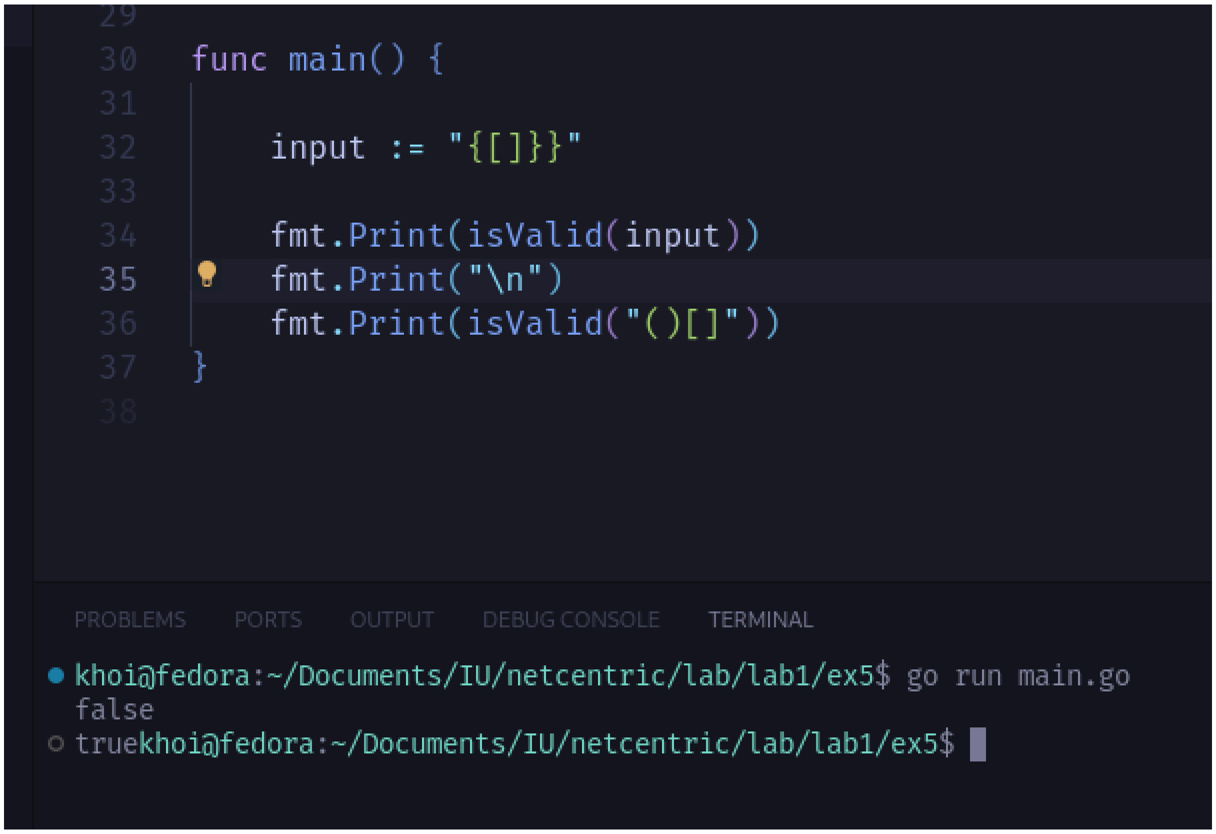
- Explanation: First I create an empty board and fill it with “.”. Then I create mines with random locations.
- In order to avoid position overlapping, I use an array to store mine coordinates and whenever a location is created it will be checked whether it is available or not.
- Then I will loop through every position, count how many bombs surround it and assign the number.

- Finally print the board.

5.
main.go

```go
package main

import "fmt"

func isValid(s string) bool {
   stack := make([]rune, 0)

   mapping := map[rune]rune{
       ')': '(',
       ']': '[',
       '}': '{',
   }

   for _, char := range s {
       if char == '(' || char == '[' || char == '{' {
           stack = append(stack, char)
       } else {
           if len(stack) == 0 {
               return false
           }
           if stack[len(stack)-1] != mapping[char] {
               return false
           }
           stack = stack[:len(stack)-1]
       }
   }
   return len(stack) == 0
}

func main() {
   input := "{[]}}"

   fmt.Print(isValid(input))
}
```

Result



```go
func main() {

    input := "{[]}}"

    fmt.Print(isValid(input))
    fmt.Print("\n")
    fmt.Print(isValid("()[]"))
}
```

```
khoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex5$ go run main.go
false
truekhoi@fedora:~/Documents/IU/netcentric/lab/lab1/ex5$
```

- In this problem I use an array and turn it in to a stack.