Net-Centric Lab 1

Student: Nguyen Duc Toan

ID: ITCSIU21112

Question 1: Hamming Distance

Q1.go:

Lab1 > Q1 > Q1.go > ...

```go
package main

import (
    "fmt"
    "math/rand"
)

var dna = []string{"A", "T", "C", "G"}

func randomDNA(size int) []string {
    DNA := make([]string, size)

    for i := range size {
        DNA[i] = dna[rand.Intn(4)]
    }
    return DNA
}

func hammingDistance(size int) int {
    var DNA1 = randomDNA(size)
    var DNA2 = randomDNA(size)

    if len(DNA1) != len(DNA2) {
        return -1
    }

    distance := 0
    for i := range DNA1 {
        if DNA1[i] != DNA2[i] {
            distance++
        }
    }

    fmt.Println("DNA1:", DNA1)
    fmt.Println("DNA2:", DNA2)
    fmt.Printf("Hamming distance: ")
    return distance
```

Main.go:

```go
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Q1: Hamming")
    var size int
    fmt.Print("Input the DNA size: ")
    fmt.Scan(&size)
    for i := 1; i <= 1000; i++ {
        fmt.Println("Pair number", i)
        fmt.Println(hammingDistance(size))
        fmt.Println("====================================")
    }
}

```

Result:

```
Pair number 993
DNA1: [T T C G T A T T C G C G A A A G T]
DNA2: [T T G A T T C C G T G G C A T G A]
Hamming distance: 11
====================================
Pair number 994
DNA1: [A T C C A G A C C A A T C G T A A]
DNA2: [T G G C T T T C G C C A C C T C G]
Hamming distance: 13
====================================
Pair number 995
DNA1: [C T A T A T C G G T T C G T A G A]
DNA2: [G T C T A T G A T T G G C C G T C]
Hamming distance: 12
====================================
Pair number 996
DNA1: [T C G A A A A C G C A T G C C C C]
DNA2: [A C A A G T C A G C C T A G T A T]
Hamming distance: 12
====================================
Pair number 997
DNA1: [T T G G G G T C C G G C G T T A T]
DNA2: [A T C A T C T G C A G C C T T A T]
Hamming distance: 8
====================================
Pair number 998
DNA1: [A T T C C G A C C G C A G T C A T]
DNA2: [A C A C G A T T A C A T C C G A T]
Hamming distance: 13
====================================
Pair number 999
DNA1: [C G C G A T A G C C A T A G G G G]
DNA2: [C A A C C G C A G G T G G A T T A]
Hamming distance: 16
====================================
Pair number 1000
DNA1: [C C T T G T C T T A T C G C C C T]
DNA2: [A A A C C A G C T T T G T C A C C]
Hamming distance: 13
====================================
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q1>
```

Question 2: Scrabble Score

Q2.go

```go
package main

var scrabbleScore = map[string]int{
    "A": 1, "E": 1, "I": 1, "O": 1, "U": 1, "L": 1, "N": 1, "R": 1, "S": 1, "T": 1,
    "D": 2, "G": 2,
    "B": 3, "C": 3, "M": 3, "P": 3,
    "F": 4, "H": 4, "V": 4, "W": 4, "Y": 4,
    "K": 5,
    "J": 8, "X": 8,
    "Q": 10, "Z": 10,
}

func calculateScarbbleScore(word string) int {
    score := 0
    for i := range word {
        score += scrabbleScore[string(word[i])]
    }
    return score
}

```

Main.go

Lab1 > Q2 > main.go > ...

```go
package main

import (
    "fmt"
    "strings"
)

func main() {
    fmt.Println("Q2: Scrabble Score")
    var words string
    fmt.Print("Type a string: ")
    fmt.Scan(&words)
    // fmt.Scanf("%s", &words) // Use for multiple words
    fmt.Println(calculateScarbbleScore(strings.ToUpper(words)))
    fmt.Println("====================================")
}

```

Result

```
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q2> go run .
Q2: Scrabble Score
Type a string: cabbage
14
==================================
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q2>
```

Question 3: Luhn

Q3.go

```go
package main

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

func doubling(number string) int {
    product, err := strconv.Atoi(number)
    if err != nil {
        // ... handle error
        panic(err)
    }
    if product*2 > 9 {
        product = product*2 - 9
    } else {
        product *= 2
    }
    return product
}
```

```go
func validating(number string) bool {
    fmt.Println("Number: " + number)
    number = strings.ReplaceAll(number, " ", "")
    if len(number) <= 1 {
        return false
    }
    total := 0
    for i := len(number) - 2; i >= 0; i -= 2 {
        total += doubling(string(number[i]))
    }
    for i := len(number) - 1; i >= 0; i -= 2 {
        digit, err := strconv.Atoi(string(number[i]))
        if err != nil {
            // ... handle error
            panic(err)
        }
        total += digit
    }
    fmt.Printf("Check sum: %d\n", total)

    return total%10 == 0
}

```

Main.go

```go
Lab1 > Q3 > main.go > ...
package main

import (
    "fmt"
)

func main() {

    fmt.Println("Q3: Luhn")
    if validating("4539 3195 0343 6467") {
        fmt.Println("The number is valid.")
    } else {
        fmt.Println("The number is invalid.")
    }
    fmt.Println("====================================")
}

```

Result

```
PS E:\IU\Senior\Net-Centric Lab\Lab1\q3> go run .
Q3: Luhn
Number: 4539 3195 0343 6467
Check sum: 80
The number is valid.
==================================
PS E:\IU\Senior\Net-Centric Lab\Lab1\q3>
```

Question 4: Minesweeper

Q4.go

Lab1 > Q4 > Q4.go > checkNeighbour

```go
package main

import (
    "fmt"
    "math/rand"
)

// create a board with random mines position
func createBoard(width int, height int, mines int) [][]string {
    board := make([][]string, height+1)
    minesCoordinates := make([][]int, mines)
    for i := range board {
        board[i] = make([]string, width+1)
    }
    for i := range minesCoordinates {
        minesCoordinates[i] = make([]int, 2)
    }
    for i := range board {
        for j := range board[i] {
            board[i][j] = ". "
        }
    }
    for i := 0; i < mines; i++ {
        x := rand.Intn(width) + 1
        y := rand.Intn(height) + 1
        for j := 0; j < i; j++ {
            if minesCoordinates[j][0] == x && minesCoordinates[j][1] == y {
                x = rand.Intn(width) + 1
                y = rand.Intn(height) + 1
                j = 0
            }
        }
        minesCoordinates[i][0] = x
        minesCoordinates[i][1] = y
        board[x][y] = "* "
    }
    return board
```

```go
var neighbourOffset = [8][2]int{
    {-1, -1}, {-1, 0}, {-1, 1},
    {0, -1}, {0, 1},
    {1, -1}, {1, 0}, {1, 1},
}

func checkNeighbour(x, y int, board [][]string) int {
    count := 0
    for i := range neighbourOffset {
        posX := x + neighbourOffset[i][0]
        posY := y + neighbourOffset[i][1]
        if posX >= 0 && posX < len(board) && posY >= 0 && posY < len(board) {
            if board[posX][posY] == "* " {
                count++
            }
        }
    }
    return count
}

func displayBoard(board [][]string) {
    for i := range board {
        for j := range board[i] {
            if board[i][j] == ". " {
                if checkNeighbour(i, j, board) == 0 {
                    board[i][j] = ". "
                } else {
                    board[i][j] = fmt.Sprintf("%d ", checkNeighbour(i, j, board))
                }
            }
        }
    }
    for i := range board {
        fmt.Println(board[i])
    }
}
```

Main.go

```go
package main

import (
    "fmt"
)

func main() {
    var width, height, mines int
    fmt.Println("Q4: Minesweeper")
    fmt.Print("Enter the width of the board: ")
    fmt.Scan(&width)
    fmt.Print("Enter the height of the board: ")
    fmt.Scan(&height)
    fmt.Print("Enter the number of mines: ")
    fmt.Scan(&mines)
    board := createBoard(width, height, mines)
    displayBoard(board)
    fmt.Println("==================================")
}

```

Result:

```
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q4> go run .
Q4: Minesweeper
Enter the width of the board: 25
Enter the height of the board: 25
Enter the number of mines: 99
[.  .  .  .  .  .  .  .  .  1  1  1  .  .  1  1  1  1  1  1  .  .  .  .  .  .  .]
[.  .  1  1  1  .  .  .  1  *  1  .  .  1  *  1  1  *  1  .  .  1  1  1  .  .]
[.  .  1  *  1  .  .  1  3  4  3  1  .  1  1  1  1  1  2  1  1  1  *  1  .  .]
[.  .  2  2  2  .  1  2  *  *  *  1  .  .  1  2  2  2  3  *  2  1  1  2  1  1]
[.  .  1  *  1  .  1  *  3  4  3  2  1  1  2  *  *  2  *  *  2  .  .  2  *  2]
[.  .  1  1  1  .  1  1  1  1  *  2  2  *  2  2  2  3  4  4  3  2  2  3  *  2]
[.  .  1  1  2  2  2  1  .  1  2  3  *  3  2  1  .  1  *  *  2  *  *  3  3  2]
[.  1  2  *  2  *  *  2  1  1  2  *  3  2  *  1  .  1  2  2  3  3  4  *  3  *]
[.  1  *  2  2  2  2  2  *  2  3  *  2  1  1  1  .  1  1  1  1  *  2  2  *  2]
[.  1  1  1  .  .  .  1  2  *  2  1  1  .  .  .  .  1  *  2  2  2  1  1  1  1]
[.  .  .  .  .  .  .  .  .  1  1  2  1  1  1  1  1  .  1  2  3  *  1  .  .  1  1]
[1  1  1  .  .  1  1  1  .  .  1  *  1  2  *  2  .  .  1  *  2  1  .  .  2  *]
[1  *  1  1  2  3  *  1  .  .  1  1  1  2  *  3  2  2  2  1  1  .  .  .  3  *]
[1  1  1  1  *  *  3  3  1  2  1  1  .  1  1  2  *  *  1  .  .  .  .  .  2  *]
[1  1  2  2  3  3  *  2  *  2  *  2  1  .  .  1  3  3  2  .  .  .  .  .  1  1]
[2  *  3  *  1  2  2  3  2  3  4  *  3  1  1  .  1  *  3  2  1  .  1  1  1  .]
[2  *  4  2  2  2  *  1  1  *  4  *  5  *  2  .  1  2  *  *  2  .  1  *  1  .]
[1  3  *  2  2  *  4  2  2  1  3  *  4  *  2  .  .  1  3  *  2  .  1  2  2  1]
[.  3  *  4  3  *  3  *  1  .  1  2  3  2  1  .  1  1  2  1  1  .  .  1  *  1]
[.  2  *  3  *  3  3  2  1  .  .  1  *  1  .  .  1  *  2  2  2  2  2  3  4  3]
[.  1  1  3  2  4  *  2  .  .  .  1  2  2  1  .  1  2  *  2  *  *  2  *  *  *]
[.  .  1  2  *  4  *  2  .  .  .  .  1  *  1  .  .  1  1  2  2  2  2  2  4  3]
[.  .  1  *  3  *  3  2  1  .  .  1  2  2  1  .  .  .  .  .  .  .  .  .  1  *]
[.  .  1  1  2  2  3  *  1  .  .  2  *  2  .  .  .  .  .  .  .  .  .  .  1  1]
[.  .  .  1  1  2  *  2  1  .  .  2  *  2  .  .  1  1  1  .  .  .  .  .  .  .]
[.  .  .  1  *  2  1  1  .  .  .  1  1  1  .  .  1  *  1  .  .  .  .  .  .  .]
==================================
```

Question 5: Matching Brackets

Q5.go

```go
package main

func isCorrect(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 || stack[len(stack)-1] != mapping[char] {
            return false
        } else {
            stack = stack[:len(stack)-1]
        }
    }
    return len(stack) == 0
}

```

Main.go

```go
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Q5: Matching Brackets")
    typeOfBracket := "{{[]}}"
    fmt.Println(typeOfBracket)
    if isCorrect(typeOfBracket) {
        fmt.Println("The brackets are matched and nested correct.")
    } else {
        fmt.Println("The brackets are matched and nested incorrect.")
    }
    fmt.Println("====================================")
}

```

Result:

```
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q5> go run .
Q5: Matching Brackets
{{[]}}
The brackets are matched and nested correct.
====================================
PS E:\IU\Senior\Net-Centric Lab\Lab1\Q5>
```