Fonctions

Exercice 1

```go
package main

import "fmt"

func add(a, b int) int {
    return a + b
}

func multiply(a, b int) int {
    return a * b
}

func greet(name string) string {
    return "Hello " + name
}

func isEven(a int) bool {
    return a % 2 == 0
}

func average(numbers []float64) float64 {
    var sum float64
    for _, number := range numbers {
        sum += number
    }
    return sum / float64(len(numbers))
}

func main() {
    fmt.Println(add(5, 3))
    fmt.Println(multiply(4, 7))
    fmt.Println(greet("Alice"))
    fmt.Println("6 is even:", isEven(6))
    fmt.Println("9 is even:", isEven(9))
    fmt.Printf("Average : %.2f\n", average([]float64{2.5, 3.7, 4.8}))
}
```

Exercice 2

```go
package main

import (
    "bufio"
    "fmt"
    "math/rand"
    "os"
    "strconv"
)
```

```
func generatePassword(length int, uppercase bool, lowercase bool, numbers bool, specials bool)
string {
    password := ""
    for i := 0; i < length; i++ {
        password += getRandomChar(uppercase, lowercase, numbers, specials)
    }
    return password
}

func getRandomChar(uppercase bool, lowercase bool, numbers bool, specials bool) string {
    numberChar := "0123456789"
    uppercaseChar := "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    lowercaseChar := "abcdefghijklmnopqrstuvwxyz"
    specialsChar := "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"

    possibleChars := ""
    if uppercase {
        possibleChars += uppercaseChar
    }
    if lowercase {
        possibleChars += lowercaseChar
    }
    if numbers {
        possibleChars += numberChar
    }
    if specials {
        possibleChars += specialsChar
    }

    return string(possibleChars[rand.Intn(len(possibleChars))])
}

func promptUserBoolean(scanner *bufio.Scanner, question string) bool {
    for {
        fmt.Print(question)
        scanner.Scan()
        switch scanner.Text() {
        case "Oui", "oui", "O", "o":
            return true
        case "Non", "non", "N", "n":
            return false
        default:
            fmt.Print("Veuillez entrer une réponse valide : ")
        }
    }
}

func promptUserInt(scanner *bufio.Scanner, question string, minimum int) int{
    var number int
    var err error
    for {
        fmt.Print(question)
        scanner.Scan()
        number, err = strconv.Atoi(scanner.Text())
        if err != nil {
            fmt.Print("Veuillez entrer un nombre valide : ")
            continue
        }
        if number < minimum {
            fmt.Print("Veuillez entrer un nombre supérieur ou égal à", minimum, " : ")
            continue
        }
        return number
    }
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    var passwordLength int
    var uppercase, lowercase, numbers, specials bool

    fmt.Println("Générateur de Mot de Passe Améliorée")
    fmt.Println("====================================\n")

    passwordLength = promptUserInt(scanner, "Entrez la longueur du mot de passe : ", 1)
    fmt.Println()
    uppercase = promptUserBoolean(scanner, "Inclure des majuscules ? (Oui/Non) : ")
    lowercase = promptUserBoolean(scanner, "Inclure des minuscules ? (Oui/Non) : ")
    numbers = promptUserBoolean(scanner, "Inclure des chiffres ? (Oui/Non) : ")
    specials = promptUserBoolean(scanner, "Inclure des caractères spéciaux ? (Oui/Non) : ")
    fmt.Println()

    if !uppercase && !lowercase && !numbers && !specials {
        fmt.Println("Vous devez choisir au moins une option.")
        return
    }

    password := generatePassword(passwordLength, uppercase, lowercase, numbers, specials)
    fmt.Println("Mot de passe généré :", password)

}
```

Exercice 3

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
)

func convertToCelsius(temperature float64, scaleFrom string) float64 {
    switch scaleFrom {
    case "Celsius":
        return temperature
    case "Fahrenheit":
        return (temperature - 32) * 5 / 9
    case "Kelvin":
        return temperature - 273.15
```

```go
    }
    return 0
}

func convertToFahrenheit(temperature float64, scaleFrom string) float64 {
    switch scaleFrom {
    case "Celsius":
        return temperature*9/5 + 32
    case "Fahrenheit":
        return temperature
    case "Kelvin":
        return temperature*9/5 - 459.67
    }
    return 0
}

func convertToKelvin(temperature float64, scaleFrom string) float64 {
    switch scaleFrom {
    case "Celsius":
        return temperature + 273.15
    case "Fahrenheit":
        return (temperature + 459.67) * 5 / 9
    case "Kelvin":
        return temperature
    }
    return 0
}

func convertTemperature(temperature string, scaleFrom string, scaleTo string) float64 {
    temp, _ := strconv.ParseFloat(temperature, 64)

    switch scaleTo {
    case "Celsius":
        return convertToCelsius(temp, scaleFrom)
    case "Fahrenheit":
        return convertToFahrenheit(temp, scaleFrom)
    case "Kelvin":
        return convertToKelvin(temp, scaleFrom)
    }
    return 0
}

func main() {
    fmt.Println("Convertisseur de Température")
    fmt.Println("----------------------------")
    fmt.Println("1. Celsius vers Fahrenheit")
    fmt.Println("2. Celsius vers Kelvin")
    fmt.Println("3. Fahrenheit vers Celsius")
    fmt.Println("4. Fahrenheit vers Kelvin")
    fmt.Println("5. Kelvin vers Celsius")
    fmt.Println("6. Kelvin vers Fahrenheit")
    fmt.Println()
    fmt.Print("Sélectionnez une option : ")

    in := bufio.NewScanner(os.Stdin)
    in.Scan()

    switch in.Text() {
    case "1":
        fmt.Print("Entrez la température en degrés Celsius : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Fahrenheit : %.2f\n", convertTemperature(temperature,
"Celsius", "Fahrenheit"))
    case "2":
        fmt.Print("Entrez la température en degrés Celsius : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Kelvin : %.2f\n", convertTemperature(temperature, "Celsius",
"Kelvin"))
    case "3":
        fmt.Print("Entrez la température en degrés Fahrenheit : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Celsius : %.2f\n", convertTemperature(temperature,
"Fahrenheit", "Celsius"))
    case "4":
        fmt.Print("Entrez la température en degrés Fahrenheit : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Kelvin : %.2f\n", convertTemperature(temperature,
"Fahrenheit", "Kelvin"))
    case "5":
        fmt.Print("Entrez la température en degrés Kelvin : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Celsius : %.2f\n", convertTemperature(temperature, "Kelvin",
"Celsius"))
    case "6":
        fmt.Print("Entrez la température en degrés Kelvin : ")
        in.Scan()
        temperature := in.Text()
        fmt.Printf("Température en Fahrenheit : %.2f\n", convertTemperature(temperature,
"Kelvin", "Fahrenheit"))
    default:
        fmt.Println("Option invalide")
    }

}
```

Exercice 4

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
```

```go
)

func createBadge(name string, title string) string {
    return strings.Repeat("*", 17) + "\n" +
        "Nom: " + name + "\n" +
        "Titre: " + title + "\n" +
        strings.Repeat("*", 17)
}

func promptParticipant(i int) (string, string) {
    var in = bufio.NewScanner(os.Stdin)
    var name, title string
    fmt.Printf("Participant %d :\n", i)
    fmt.Print("Entrez le nom : ")
    in.Scan()
    name = in.Text()
    fmt.Print("Entrez le titre : ")
    in.Scan()
    title = in.Text()

    return name, title
}

func main() {
    fmt.Println("Générateur de Badges pour un Événement")
    fmt.Println("--------------------------------------\n")

    var in = bufio.NewScanner(os.Stdin)
    fmt.Print("Combien de participants sont présents à l'événement ? : ")
    in.Scan()
    nbParticipants, _ := strconv.Atoi(in.Text())

    var badges []string
    for i := 1; i <= nbParticipants; i++ {
        name, title := promptParticipant(i)
        fmt.Println()
        badges = append(badges, createBadge(name, title))
    }

    fmt.Println("\nBadges générés pour l'événement :")
    for _, badge := range badges {
        fmt.Println(badge)
    }
}
```