Tableaux et Tranches

Exercice 1

- Exécutez d'abord le programme suivant tel quel.
- Ensuite, changez les déclarations de tableaux en déclarations de tranches (slices).
- Observez si quelque chose change ou non (en surface :)).

```
package main

import "fmt"

func main() {
    names := [3]string{"Einstein", "Tesla", "Shepard"}
    distances := [...]int{50, 40, 75, 30, 125}
    data := [5]byte{'H', 'E', 'L', 'L', 'O'}
    ratios := [1]float64{3.14145}
    alives := [...]bool{true, false, true, false}
    zero := [0]byte{}

    fmt.Printf("names    : %T %q\n", names)
    fmt.Printf("distances: %T %d\n", distances)
    fmt.Printf("data     : %T %d\n", data)
    fmt.Printf("ratios   : %T %.2f\n", ratios[0])
    fmt.Printf("alives   : %T %t\n", alives)
    fmt.Printf("zero     : %T %d\n", zero)
}
```

Exercice 2

- Divisez la chaîne de caractères `namesA` et obtenez une tranche (slice).
- Triez toutes les tranches.
- Comparez si les tranches sont égales ou non

```
func main() {
    // namesA := "Da Vinci, Wozniak, Carmack"
    // namesB := []string{"Wozniak", "Da Vinci", "Carmack"}
}
```

Indices :

- La fonction `strings.Split` divise une chaîne de caractères et renvoie une tranche (slice) de chaînes de caractères.
- La fonction `sort.Strings` trie une tranche (slice) de chaînes de caractères.
- Pour comparer des tranches (slices) : commencez par vérifier si leur longueur est la même ou non ; seulement ensuite comparez-les.

Sortie attendue :

```
Ils sont égaux.
```

Exercice 3

- Créez les tranches (slices) vides suivantes :

    - Garnitures de pizza
    - Heures de départ
    - Années de diplomation des étudiants
    - États marche/arrêt des lumières dans une pièce
- Ajoutez-leur quelques éléments (faites preuve de créativité !).
- Affichez toutes les tranches (slices).

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // Votre code ici
}
```

Indices :

- Pour les heures de départ, utilisez le type time.Time. Consultez sa documentation (https://gobyexample.com/time).

```
now := time.Now() -> Vous donne l'heure actuelle
now.Add(time.Hour*24) -> Vous donne un time.Time 24 heures après now
```

- Pour les années de diplomation, vous pouvez utiliser le type int.

Résultat attendu (Votre résultat peut varier, le mien ressemble à ceci) :

```
pizza       : [pepperoni onions extra cheese]

graduations : [1998 2005 2018]

departures  : [2019-01-28 15:09:31.294594 +0300 +03 m=+0.000325020
2019-01-29 15:09:31.294594 +0300 +03 m=+86400.000325020
2019-01-30 15:09:31.294594 +0300 +03 m=+172800.000325020]

lights      : [true false true]
```

Exercice 4

Écrivez un programme qui ajoute des éléments à une tranche (slice) 10 millions de fois dans une boucle. Observez comment la capacité de la tranche (slice) change.

- Créez une tranche (slice) vide.
- Effectuez une boucle 10 millions de fois.
- À chaque itération : ajoutez un élément à la tranche (slice).
- Affichez la longueur (length) et la capacité (capacity) de la tranche (slice) uniquement lorsque sa capacité change.

BONUS : Affichez également le taux de croissance de la capacité.

```
package main
```

```go
func main() {
    // Votre code ici
}
```

Résultat attendu :

```
len:0 cap:0 growth:NaN
len:1 cap:1 growth:+Inf
len:2 cap:2 growth:2.00
... et ainsi de suite...
```