Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Devoir 3 (6%)

Date de remise: le 10 Avril 2018 avant 23:00 sur le campus virtuel

Question 1. **[2.5 points]**

Définir une fonction AbsDiff permettant de calculer la différence absolue entre éléments d'un slice de nombre réels (float32). Cette fonction retourne un slice et un code d'erreur dans le cas où les listes en entrée ne sont pas de la même longueur.

```
func AbsDiff(sliceA, sliceB []float32)
                          (res []float32, err error)
```

Si les deux slices ne sont pas de longueur égale, on imprime:
`Slices are not the same length`.

Créer une fonction `main` demandant à un utilisateur d'entrer une nouvelle liste, on imprime alors le résultat en utilisant les deux dernières listes entrées (dans le cas de la première slice, la slice précédente est égale à la slice entrée). Une boucle doit permettre d'entrer de nouvelles slices de façon répétitive.

```
Previous slice: []
Enter another slice of floating point numbers (Anything else to end
slice)
3.2 -6.77 42 -0.9
Result: [0.0 0.0 0.0 0.0]
q to quit (Anything else to continue): c

Previous slice: [3.2 -6.77 42 -0.9]
Enter another slice of floating point numbers (Anything else to end
slice)
5.4 6 7.8 -10
Result: [2.2 12.77 34.2 9.1]
```

```
q to quit (Anything else to continue): c

Previous slice: [5.4 6 7.8 -10]
Enter another slice of floating point numbers (Anything else to end
slice)
17.5 1123.98 0.001

Slices are not the same length
q to quit (Anything else to continue): q
```

Créer une nouvelle version de cette fonction, cette fois-ci avec un paramètre entier permettant de spécifier la stratégie à adopter dans le cas de slices de taille différentes.

```
func AbsDiff2(sliceA, sliceB []float32, version int)
                              (res []float32, err error)
```

a) Si version est égal à 0, alors la fonction se comporte comme la version originale en retournant une erreur.
b) Si version est négatif, alors les éléments manquants de la liste la plus petit sont supposés être 0. Le résultat est de la même longueur que la liste la plus longue.
c) Si version est positif, alors les éléments en surplus sont ignorés. Le résultat est de longueur égal à la liste la plus courte.

Question 2. **[3.5 points]**

1. Créer le type `struct Bread` avec les attributs suivants:
   - Un `string` pour le nom (`name`)
   - Un map avec une clé de type `string` et une valeur de type `Item` pour les `ingredients`
     - Avec `type Item struct {weight int}`
   - Un `float32` pour le poids (`weight`) en kilogrammes
   - Une structure `struct baking` contenant les informations de cuisson. Cette structure doit avoir trois attributs: `bakeTime` et `coolTime` en minutes, la `temperature` en Celcius, tous des `int`.
2. Créer une interface `Baker` avec les deux méthodes suivantes:
   - `shoppingList` acceptant une liste d'ingrédients sous forme d'un map de `string`:`Item` et retournant deux listes d'ingrédients comme map de `string`:`Item`.
   - `printBakeInstructions` sans arguments et sans valeurs de retour.
   - `printBreadInfo` sans arguments et sans valeurs de retour.
3. Concevoir les fonctions suivantes

- `NewBread` retournant un pointeur à `Bread` avec l'attribut `name` à "Whole Wheat", le champ `ingredients` attribué aux paires clé/valeur suivantes `"whole wheat flour":{500}, "yeast":{25}, "salt":{25}, "sugar":{50}, "butter":{50}, "water":{350}`. Le temps de cuisson est de 2 heures à 180 degrés Celsius avec un temps de refroidissement de 1 heure. Calculer le poids total comme étant la somme du poids de tous les ingrédients.
- `NewBreadVariation` acceptant un nouveau nom et deux maps `string`:`Item` spécifiant les ingrédients à ajouter et les ingrédients à retirer.

4. Définir les méthodes de l'interface `Baker` pour un pointeur au type `Bread`
   - La méthode `shoppingList` à laquelle est donnée la liste des items disponibles, sous la forme d'un un map de `string`:`Item`. Le résultat est alors la liste des ingrédients manquants (la liste d'épicerie) et la nouvelle liste des items diponibles, c'est-à-dire la liste d'entrée à laquelle sont soustraits les ingrédients utilisé pour réaliser la recette.
   - La méthode printBakeInstructions imprimant la température de caisson et la durée en minutes.
   - La méthode printBakeInfo imprimant nom, ingredients et poids.
5. Définir une fonction `main` créant deux `Bread`s placés dans un slice de `Baker`: un pain standard "whole wheat" et un "Sesame" avec farine moitié blé entier et moitié farine blanche. Puis obtenez une liste d'ingrédients à acheter en supposant que vous avez 5kg de farine de blé entier, 500g de sel et 1Kg de sucre. Imprimer la liste des ingrédients à acheter ainsi que les instructions de cuisson.

   .

Exemple de sortie:

```
Whole Wheat bread
map[butter:{50} water:{350} whole wheat flour:{500} yeast:{25} salt:
{25} sugar:{50}]
Weight 1.000 kg

Sesame bread
map[salt:{25} sugar:{50} butter:{50} water:{350} white flour:{200}
sesame:{50} whole wheat flour:{250} yeast:{25}]
Weight 1.000 kg


Shopping List:
map[butter:{50} water:{350} white flour:{200} sesame:{50} yeast:{25}]

Baking Instructions:
Bake at 180 Celsius for 120 minutes and let cool for 60 minutes.
Bake at 180 Celsius for 120 minutes and let cool for 60 minutes.
```

Question 3. **[2 points]**

La fonction `RandomArray` permet de générer un tableau de nombres aléatoires.

```
func RandomArray(len int) []float32 {
    array := make([]float32, len)
    for i := range array {
        array[i] = rand.Float32()
    }

    return array
}
```

Soit le programme (incomplet) suivant permettant le traitement concurrent d'un millier de tableaux.

```
func main() {
   rand.Seed(100) // use this seed value

   out := make(chan float32)
   defer close(out)

   for i := 0; i<1000 ; i++ {
      a:= RandomArray(2*(50+rand.Intn(50)))
      go Process(a,out)
   }

   // ******
   // read here the results of the processing
   // and sum these results

   fmt.Println(sum)
}
```

La fonction `Process` effectuant le traitement doit procéder comme suit :

- Séparer le tableau en deux parties égales (les N/2 premiers éléments et les N/2 derniers éléments)
- Appeler la fonction AbsDiff avec ces deux demi-tableaux
- Effectuer la somme des éléments du tableau résultant
- Retourner cette somme via un channel

Écrire la fonction `Process` et compléter la fonction `main`