1 Programming M2 2017-2018

1.1 Setup

Pour l'examen final, il **faudra** mettre le code sous:

- `$HOME/M_<username>/go/src/uca.fr/<username>`

Pour s'assurer que tout le monde a la bonne configuration:

```
$> export GOPATH=$HOME>M_username/go
$> go get github.com/master-pfa-info/exams/2017-2018/m2/setup-pfa-exam
$> $GOPATH/bin/setup-pfa-exam -init
```

(remplacer `M_username` par le bon nom de repertoire.)

À la fin de l'examen, il faudra lancer:

```
$> $GOPATH/bin/setup-pfa-exam -save
```

et envoyer le fichier résultant à: `binet@cern.ch`.

1.2 ROT-13 (5 pts)

Nous avons vu pendant le cours que César avait inventé une méthode pour chiffrer ses messages importants. Cette procedure est maintenant appelée *"Le chiffre de César"*, ou bien *"déplacer de 13 places"* (*ie:* `ROT13`).

Pendant le cours, nous avons vu comment créer un `io.Reader` qui chiffre un message avec cette technique.

Pour cet exercice, vous devrez implanter un `io.Writer` qui chiffre les messages *via* cette technique.

- Créer un `package rot13` sous `$GOPATH/src/uca.fr/<username>/rot13`
- Créer une commande (`package main`) sous `$GOPATH/src/uca.fr/<username>/rot13/cmd/rot13`

Le paquet `rot13` devra contenir l'implantation de l'`io.Writer` qui réalise le chiffrement. La commande `rot13` devrait ressembler a:

```
package main

import (
    "io"
    "os"

    "uca.fr/<username>/rot13"
)

func main() {
    w := rot13.NewWriter(os.Stdout)
    _, err := io.Copy(w, os.Stdin)
```

```
    if err != nil {
        panic(err)
    }
}
```

1.3 Multiples (5 pts)

Si on liste tous les entiers naturels plus petits que 10 qui sont des multiples de 3 ou 5, on obtient 3, 5, 6 et 9. La somme de ces multiples est 23.

Trouvez la somme de tous les multiples de 3 ou 5 plus petits que 1000.

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/mult`.

1.4 Fibonacci (5 pts)

Chaque nouveau terme de la suite de Fibonacci est généré en ajoutant les 2 termes précédants. En partant de 1 et 2, les 10 premiers termes sont:

```
1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
```

En considérant les termes de la suite de Fibonacci dont les valeurs n'excèdent pas 4 millions, trouvez la somme des termes pairs.

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/fib`.

1.5 Ant and seeds (5 pts)

Une fourmi marche aléatoirement sur une grille `5x5`. La marche démarre dans le carré central. À chaque étape, la fourmi se déplace vers un carré adjacent de manière aléatoire, sans jamais sortir de la grille. Elle n'a donc que 2, 3 ou 4 mouvements possibles à chaque étape selon sa position.

Au debut de la marche, une graine est placée sur chaque carré de la rangée la plus basse. Quand la fourmi ne porte pas de graine et atteint un carré de la rangée la plus basse qui contient une graine, elle prend la graine avec elle. La fourmi dépose la graine dès qu'elle rencontre un carré vide de la rangée la plus haute.

Quel est le nombre attendu d'étapes pour que toutes les graines soient déposées dans la rangée la plus haute? Donnez votre réponse arrondie à 6 décimales.

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/ants`.

1.6 Black & White (5 pts)

Une fourmi se déplace sur une grille de cellules de couleur blanche ou noire. La fourmi est toujours orientée selon l'un des 4 points cardinaux (N,S,E,O) et se déplace de cellule en cellule selon les règles suivantes:

- si elle est sur une cellule noire, elle change la couleur en blanc, effectue une rotation de +90 degrees dans le sens trigonométrique et avance d'une cellule,
- si elle est sur une cellule blanche, elle change la couleur en noir, effectue une rotation de -90 degrees dans le sens trigonométrique et avance d'une cellule.

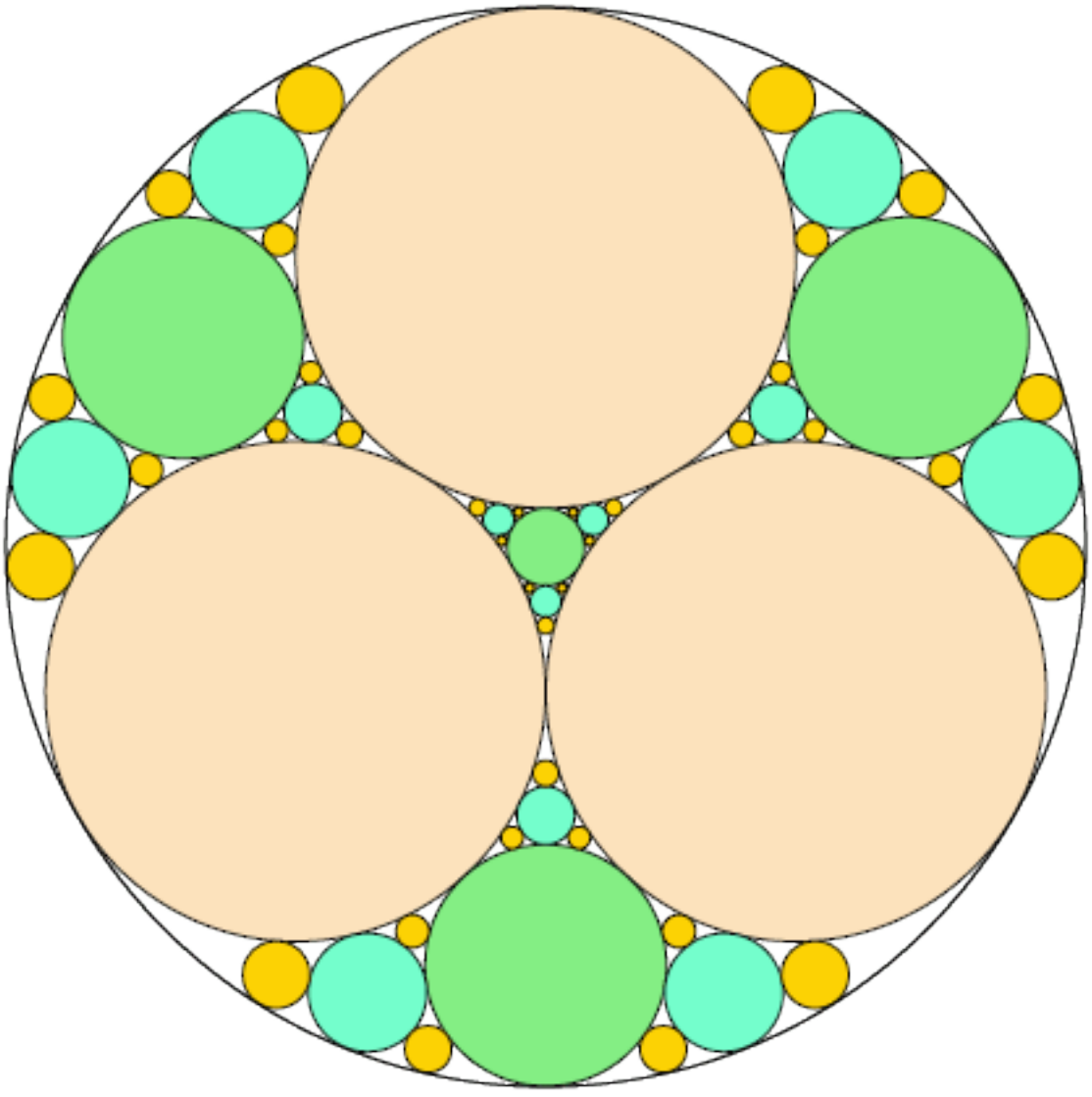
En commançant avec une grille entièrement blanche, combien de cellules sont noires apres `10e6` déplacements de la fourmi?

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/bw`.

1.7 Iterative circle packing (5 pts)

Trois cercles de rayons égaux sont places à l'intérieur d'un cercle plus grand tels que chaque paire de cercles est tangent à l'autre, et aucun des cercles intérieurs ne se chevauchent.

Il y a 4 *"trous"* qui peuvent être remplis de manière itérative avec d'autres cercles tangents.



À chaque itération, un cercle (le plus grand possible) est placé dans chaque trou, ce qui crée d'autres trous pour l'itération suivante. Après 3 itérations (voir figure), il y a 108 trous et la fraction de l'aire non couverte par les cercles est 0.06790342, arrondie à 8 decimales.

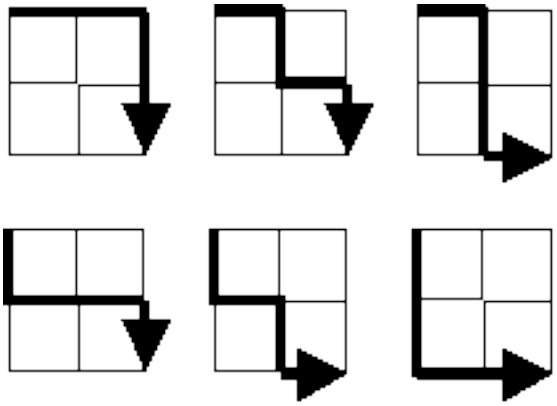
Quelle est la fraction de l'aire non couverte par des cercles apres 10 itérations? Donnez votre réponse arrondie à 8 decimales en utilisant le format`x.xxxxxxxx`.

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/circles`.

Hint: `https://en.wikipedia.org/wiki/Apollonian_gasket`

1.8 Lattice paths & concurrency (5 pts)

En commançant dans le coin superieur gauche d'une grille `2x2`, et en étant seulement autorisé à des mouvements vers la gauche ou vers le bas, il y a exactement 6 routes possibles vers le coin inférieur droit.



Combien de ces routes y a-t-il avec une grille `10x10` ?

Placez votre code sous: `$GOPATH/src/uca.fr/<username>/lattice`.

Hint: utilisez `"sync".WaitGroup`, des `goroutines` et `"sync".RWMutex`.

1.9 Mandelbrot & concurrency (5 pts)

Le programme sous `uca.fr/<username>/mbrot` génère une fractale:

```
$> cd $GOPATH/src/uca.fr/<username>/mbrot
$> go build && ./mbrot
```

Trouvez 3 méthodes pour améliorer les performances de ce programme, en utilisant des `goroutines`, des `channels` et/ou `sync.WaitGroup` (dans le package `"sync"`).

Chacune de ces 3 méthodes sera implantée dans une fonction dédiée:

- `create1`,
- `create2`,
- `create3`.

Placez votre code (contenant les 3 méthodes) sous: `$GOPATH/src/uca.fr/<username>/mbrot`.

Placez dans votre code (dans un commentaire) les mesures de temps obtenues: - pour le code originel, - pour le code avec `create1`, - pour le code avec `create2`, - pour le code avec `create3`.

Assurez vous que l'image résultante est une fractale valide (s'en assurer à l'œil suffira).