UNIVERSITÉ DE NANTES

PREMIER SEMESTRE, 2014-2015

IUT - Département d'informatique

UNIVERSITÉ DE NANTES

Devoir Surveillé - Algèbre

Durée : quatre-vingts minutes / cent sept en cas de tiers temps

Exercice 1.
On considère l'ensemble E des matrices à coefficients réels de la forme :

$$\begin{pmatrix} a & 0 \\ b & 0 \end{pmatrix}, \quad a \in \mathbb{R} \setminus \{0\},\ b \in \mathbb{R}$$

muni du produit des matrices noté $\otimes$.

a. Montrez que $\langle E, \otimes \rangle$ a une structure de magma.

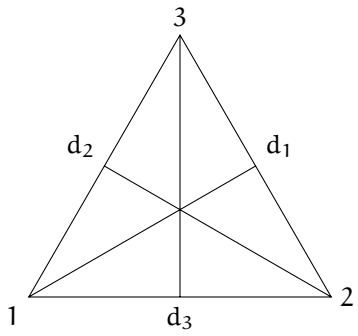
b. Est-ce que $\langle E, \otimes \rangle$ a des éléments neutres à droite ?

c. Est-ce que $\langle E, \otimes \rangle$ a des éléments neutres à gauche ?

d. Soit e un élément neutre à droite. Montrer que tout élément de E possède un inverse à gauche pour cet élément neutre, i.e.

$$(\forall g \in E)(\exists h \in E)(hg = e)$$

Exercice 2.
On considère un triangle équilatéral. On note 1, 2 et 3 ses sommets, d_1 la médiane issue de 1, d_2 celle issue de 2 et d_3 celle issue de 3 et O leur intersection.



On note f_4 la rotation de centre O et d'angle 0°, f_5 la rotation de centre O et d'angle 120°, f_6 la rotation de centre O et d'angle $-120°$ et f_1, f_2 et f_3 les symétries orthogonales d'axes respectifs d_1, d_2 et d_3.

a. On peut résumer l'action de ces transformations sous la *forme* d'une matrice :

$$\begin{pmatrix} 1 & 2 & 3 \\ f_i(1) & f_i(2) & f_i(3) \end{pmatrix}$$

Par exemple, pour f_2 :

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

Donnez les 5 autres matrices des cinq autres transformations.

b. On note E l'ensemble de ces transformations. Dressez la table de Pythagore de $\langle E, \circ \rangle$.

c. Quelle est la structure algébrique la plus « riche » de $\langle E, \circ \rangle$: magma, monoïde, groupe, anneau, demi-anneau, corps ?

d. Observez bien les représentations matricielles des éléments de E : est-ce qu'elles vous rappellent une notion étudiée en cours ?

Exercice 3.
Soit $A = \mathbb{R} \setminus \{-1, 0\}$. On considère les fonctions suivantes de A dans A :

$$f_1(x) = x \qquad f_2(x) = \frac{1}{x} \qquad f_3(x) = \frac{1}{1-x} \qquad f_4(x) = 1 - x \qquad f_5(x) = \frac{x-1}{x} \qquad f_6(x) = \frac{x}{x-1}$$

Est-ce que $E = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ est un groupe pour la composition des fonctions ?

Exercice 4.
Dans la bibliothèque `Data.Monoid` de Haskell est définie la classe de type `Monoid` de la façon suivante :

Haskell
```haskell
class Monoid a where
    mempty  :: a
    mappend :: a -> a -> a
```

Voici un extrait de la documentation :

Haskell
```
The class of monoids. Instances should satisfy the following laws:
    mappend mempty x = x
    mappend x mempty = x
    mappend x (mappend y z) = mappend (mappend x y) z
```

a. Quel est le lien entre la classe `Monoid` et la structure de monoïde étudiée en cours ? Expliquez.

b. Voici une instance possible de cette classe :

Haskell
```haskell
instance Monoid Int where
        mempty = 0
        x `mappend` y = x + y
```

i. Si l'on avait remplacé `+` par `*`, qu'aurait-on mis à droite de `mempty` ? Expliquez.

ii. On définit la fonction :

Haskell
```haskell
mconcat :: (Monoid a) => [a] -> a
mconcat = foldl mappend mempty
```

Expliquez ce que donne `mconcat ([5,6,10] :: [Int])`. On rappelle que `::` se lit « ...qui a pour type... ». On précise donc ainsi que `[5,6,10]` est une liste d'entiers.

c. Peut-on faire de `String` une instance de `Monoid` ?

Haskell
```haskell
instance Monoid String where
        mempty  = ???
        mappend = ???
```

d. Pouvez-vous définir `mempty` et `mappend` en utilisant uniquement `mconcat` ? Expliquez.

e. Une fonction `f :: a -> b` entre deux monoïdes `a` et `b` est un *morphisme* de monoïde si elle préserve la structure de monoïde :

Haskell
```
f (mempty :: a)   = mempty :: b
f (x `mappend` y) = f x `mappend` f y
```

Donnez un exemple bien connu de morphisme entre `String` et `Int` considérés comme des instances de `Monoid`.

f. On définit une classe `Groupe` pour définir des...groupes :

Haskell
```
class (Monoid g) => Groupe g where
    sym   :: g -> g
```

ainsi qu'un type `Complexe` qui représente...les nombres complexes :

Haskell
```
data Complexe = Double ::: Double
```

On fait de `Complexe` une instance de `Show` :

Haskell
```
instance Show Complexe where
    show (r:::i) = (show r) ++ " + " ++ (show i) ++ "i"
```

Par exemple :

Haskell
```
> let z = 3:::5
> z
3.0 + 5.0i
```

i. Faites alors de `Complexe` une instance de `Monoid`.

ii. À votre avis, quel est le rôle de `sym` ? Faites alors de `Complexe` une instance de `Groupe`.

iii. On définit une classe `EV` pour représenter des structures d'espaces vectoriels :

Haskell
```
class (Groupe v) => EV v k where
    (^.^) :: k -> v -> v
```

Complétez alors l'instanciation suivante :

Haskell
```
instance EV Complexe Double where
    (^.^) k  (r:::i)   = ?????
```

iv. Remplacez les trois séries de `?????` par leurs valeurs :

Haskell
```
> let z = 3:::5
> let z' = 2:::(-4)
> sym z
?????
> z `mappend` z'
?????
> (3::Double) ^.^ z
?????
```


FEUILLE DE RÉPONSE À SUIVRE

Prénom: _____ NOM: _____ Groupe: _____

Exercice 1.

a.

b.

c.

d.

Exercice 2.

a.

$\left(\quad\right)$ $\left(\quad\right)$ $\left(\quad\right)$

$\left(\quad\right)$ $\left(\quad\right)$

b.

$\circlearrowright$	f_1	f_2	f_3	f_4	f_5	f_6
f_1						
f_2						
f_3						
f_4						
f_5						
f_6						

c.

d.

Exercice 3.

Exercice 4.

a.

b. i.

ii.

iii.

c.

Haskell

```haskell
instance Monoid String where
        mempty  =
        mappend =
```

d.

e.

f.

i.

ii.

iii.

Haskell

```haskell
instance EV Complexe Double where
    (^.^) k  (r:::i)   =
```

Haskell

```haskell
> let z = 3:::5
> let z' = 2:::(-4)
> sym z


> z `mappend` z'


> (3::Double) ^.^ z

.
```