Université Grenoble Alpes **UFR IM^{2}AG**
DU ISN-C2.3 **Année 2019-20**

Langages et compilation : sémantique statique

EXERCICES (1)

Exercice 1.

En utilisant les règles formelles de sémantique statique, prouvez que le programme suivant est "correct" :

```
var x1 Entier ; var x2 Entier ; var x3 Booleen
x1 := 3 ;
tantque not x3
     x1 := x2 + 1 ;
     x3 := x3 et true
```

Exercice 2

Complétez la syntaxe abstraite et la sémantique statique des expressions du langage `while` en ajoutant les opérateurs suivants :

1. opérateurs de comparaison `<` et `=`
2. opérateurs "`e1 ? e2 : e3`" du langage C : si `e1` vaut vrai alors le résultat de l'expression est `e2`, sinon le résultat est `e3`.

Exercice 3

On étend le langage `while` en ajoutant le type `Réel`. Complétez la sémantique statique des *expressions* (relation $\longrightarrow e$) dans les deux cas suivants :

1. La conversion d'un entier vers un réel est implicite (donc toujours autorisée) ;
2. La conversion d'un entier vers un réel ne peut se faire que si l'on utilise dans le programme le nouvel opérateur `int2real (e)` qui transforme l'expression entière `e` en une expression réelle. On fournira une règle de sémantique statique pour ce nouvel opérateur.

Exercice 4

Complétez la syntaxe abstraite et la sémantique statique des commandes du langage `while` en ajoutant les instructions suivants :

1. `repeat c until e`
2. `for i in e1 .. e2 c`. On ditinguera deux cas :
   — l'instruction `for` "déclare" la variable `i` (comme en Ada)
   — l'instruction `for` "ne déclare pas" la variable `i` (qui doit donc être déclarée au préalable)