Méthodes formelles de vérification (MF)
Test
(Correction)

Exercice 1 :

On veut encoder le type des tables avec des clés de type `K` et valeurs de type `T`.

1. Définir le type inductive `Table[K,T]` avec deux constructeurs, le premier représentant la table vide, et le deuxième représentant une table au quelle on ajoute un nouveau mapping `k -> v`.

   Correction :

```
Inductive Table[K,V] :=
| Empty : Table[K,V]
| Add   : K x V x Table[K,V] -> Table[K,V]
```

2. On considère le type `Option[T]` donné en bas où les valeurs sont soit le constructeur `Some` avec une valeur de type `T`, où l'absence de valeur avec le constructeur `None`.

```
Inductive Option[T] =:
| None : Option T
| Some : T -> Option[T]
```

   Donner une fonction `Get : Table[K,T] -> K -> Option[T]` qui renvoie la valeur stockée avec la clé donnée si elle existe dans la table, où `None` sinon.

   Correction :

```
Get(Empty, k) = None
Get(Add(k',v,t), k) = if k = k' then Some(v) else Get(t, k)
```

3. Donner une fonction `Insert : Table[K,T] -> K -> T -> Table[K,T]` qui ajoute la valeur associée a la clé donnée, si elle n'existée pas déjà dans le tableau d'origine.

   Correction :

```
Insert(t, k, v) = if Get(t,k) = None then Add(k,v,t) else t
```

Exercice 2 :

On considere deux fonctions :

`GetAllKeys : Table[K,T] -> List[K]`, et
`GetAllVals : Table[K,T] -> List[T]`

qui renvoient toutes les clés dans une liste, et toutes les valeurs respectivement.

1. Donner la spécification formelle de ces fonctions (vous pouvez utiliser la fonction `Get`). Donner une implémentation récursive de `GetAllVals` et prouver qu'elle satisfait la specification donnée.

   Correction :

```
GetAllVals(Empty) = []
GetAllVals(k,v,t) = v@GetAllVals(t))

GetAllVals_Spec(t,l) := GetAllVals(t) = l ⇒
                          (∀ v, (∃ k, Get(t,k) = v)  ⟺  Mem(l,v) = true)
```

La condition de multiplicité des valeurs n'est pas requise (mais on peut ajouter une spécification en utilisant les MultiSets.)

Démonstration. Induction sur `t`.

– Cas de base : `t = Empty`.

  `(∀ v, (∃ k, Get(Empty,k) = v)  ⟺  Mem([],v) = true)` ✓

  par la definition de `Get` et `Mem`.

– Cas inductif : `t = Add(k',v,t')`, où l'hypothèse inductive est :

  `HI: (∀ v, (∃ k, Get(t,k) = v)  ⟺  Mem(l,v) = true)`

  On doit prouver que si : `GetAllValues(Add(k,v,t)) = l'` alors

  `(∀ v', (∃ k, Get(Add(k',v,t), k) = v')  ⟺  Mem(l',v') = true)`

  On a par la définition de `GetAllValues` que `l' = v@GetAllValues(t)`, et alors on peut réécrire `l'`

  `(∀ v', (∃ k, Get(Add(k',v,t), k)) = v'  ⟺  Mem(v@GetAllValues(t),v') = true)`

  On a deux cas à considérer :

  1. `v` $=$ `v'`. On sait que

     `Mem(v@GetAllValues(t),v) = true`

     On choisit `k` $=$ `k'` pour avoir

     `Get(Add(k,v,t),k') = v`

     et conclure. ✓

  2. `v` $\neq$ `v'`. Par la définition de `Mem` on a

     `Mem(v@GetAllValues(t),v') = Mem(GetAllvalues(t),v')`

     et on conclue par l'hypothèse inductive. ✓

□

2. Donner la spécification formelle des fonctions `Get` et `Insert` considérés ci-dessus, et prouver leur correction.

Correction : Exercice (facil).

Exercice 3 :

Prouvez que les triplets de Hoare suivants sont valides, ou trouvez un contre-exemple s'ils ne le sont pas et ensuite donnez des triplets corrects.

1. $\{x = 2\}$ `x := 3` $\{x = 3\}$

2. $\{x = 2\}$ `x := x + 1` $\{x = 3\}$

3. $\{y = 2\}$ `x := y` $\{x = 2\}$

4. $\{y > 0\}$ `x := y; y := -1` $\{x > 0\}$

5. $\{\text{true}\}$ `if true then y else y := 2 * x; x := y - 1` $\{x > 0\}$

6. $\{y > 0\}$ `if y > 0 then x := y else x := -y` $\{x > 0\}$

7. $\{\text{true}\}$ `if y > 0 then x := y else x := -y` $\{x > 0\}$

Correction :

1.
$$\dfrac{(x=2)\Rightarrow(3=3) \qquad \dfrac{}{\{3=3\}\ \mathtt{x := 3}\ \{x=3\}}\mathtt{aff} \qquad (x=3)\Rightarrow(x=3)}{\{x=2\}\ \mathtt{x := 3}\ \{x=3\}}\mathtt{imp}$$

2.
$$\dfrac{(x=2)\Rightarrow(x+1=3) \qquad \dfrac{}{\{x+1=3\}\ \mathtt{x := x+1}\ \{x=3\}}\mathtt{aff} \qquad (x=3)\Rightarrow(x=3)}{\{x=2\}\ \mathtt{x := x+1}\ \{x=3\}}\mathtt{imp}$$

3.
$$\dfrac{}{\{y=2\}\ \mathtt{x := y}\ \{x=2\}}\mathtt{aff}$$

4.
$$\dfrac{\dfrac{}{\{y>0\}\ \mathtt{x := y}\ \{x>0\}}\mathtt{aff} \qquad \dfrac{}{\{x>0\}\ \mathtt{y := -1}\ \{x>0\}}\mathtt{aff}}{\{y>0\}\ \mathtt{x := y; y := -1}\ \{x>0\}}\mathtt{seq}$$

5. Pas vrai si $\mathtt{x} = -\mathtt{1}$ initialement. Correction :

$$\{x>0\}\mathtt{if\ true\ then\ y\ else\ y := 2*x; x := y-1}\{x>0\}$$

6.
$$\dfrac{\dfrac{}{\{y>0\}\ \mathtt{x := y}\ \{x>0\}}\mathtt{aff\ (then)} \qquad \dfrac{(y>0\wedge y\leqslant 0)\Rightarrow(x>0) \qquad \dfrac{}{\{x>0\}\ \mathtt{y := -1}\ \{x>0\}}\mathtt{aff} \qquad (x>0)\Rightarrow(x>0)}{\{y>0\wedge y\leqslant 0\}\ \mathtt{y := -1}\ \{x>0\}}\mathtt{imp\ (else)}}{\{y>0\}\ \mathtt{if\ y>0\ then\ x := y\ else\ x := -y}\ \{x>0\}}\mathtt{cond}$$

7. Exercice (voir 5).