Méthodes formelles de vérification (MFVerif)
Test

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`.
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.
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 déjà pas dans le tableau d'origine.

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 `GetAllValues` et prouver qu'elle satisfait la specification donnée.
2. Donner la spécification formelle des fonctions `Get` et `Insert` considérés ci-dessus, et prouver leur correction.

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