IFT 2015
TP 3
(15 points)
4 avril 2014

Vous allez écrire un programme qui effectue une simulation "*Jumping Leprechauns*". Cette simulation implique un nombre arbitraire n de leprechauns, numérotés de 1 à n. Chaque leprechaun possède un chaudron rempli de g_i pièces d'or. Au début de la simulation, le chaudron de chaque leprechaun est rempli d'un million de pièces d'or ($g_i = 1\,000\,000$ pour $i = 1, 2, ..., n$). De plus, la simulation tient également, pour chaque leprechaun i, une position à l'horizon, représentée par un nombre réel x_i, et initialisée à $x_i = 0$ pour $i = 1, 2, ..., n$. À chaque itération de la simulation, les leprechauns agissent en ordre de 1 à n. L'action d'un leprechaun débute avec le calcul de sa nouvelle position à l'horizon, déterminée par

$$x_i \quad = \quad x_i + rg_i,$$

où r est un nombre réel aléatoire compris entre -1 et 1. Le leprechaun i vole alors la moitié de l'or ($\lceil$plafond$\rceil$) du leprechaun le plus proche de sa nouvelle position et ajoute cet or à son chaudron. Lorsqu'un leprechaun n'a plus d'or dans son chaudron, il disparait. Vous devez écrire un programme qui effectue autant d'itérations de cette simulation qu'il faut jusqu'à qu'il ne reste plus que deux leprechauns, pour un nombre initial n arbitraire de leprechauns. Vous devez utiliser deux structures de données de type abstrait (ADT) *sorted map*, implémentées dans **deux structures d'arbres** (différentes ou identiques), pour tenir respectivement les paires $(i,\, x_i)$ et $(x_i,\, g_i)$. Alors que les itérations sur les leprechauns devraient se faire sur la première structure, la recherche des leprechauns les plus près d'une nouvelle position devraient se faire dans la deuxième *sorted map*. Afin d'optimiser l'algorithme, le noeud correspondant à un leprechaun devrait être supprimé dans les deux structures lorsque celui-ci n'a plus d'or.

Afin de visualiser l'évolution des leprechauns, imprimez le format exact des lignes suivantes chaque fois qu'un leprechaun disparait, ainsi que les 2 gagnants et le temps d'exécution de la simulation. Par exemple, pour une simulation avec 10 leprechauns, votre output devrait ressembler à ceci :

```
Le leprechaun 9 disparait ! Il reste 9 leprechauns.
Le leprechaun 5 disparait ! Il reste 8 leprechauns.
Le leprechaun 7 disparait ! Il reste 7 leprechauns.
Le leprechaun 10 disparait ! Il reste 6 leprechauns.
Le leprechaun 6 disparait ! Il reste 5 leprechauns.
Le leprechaun 2 disparait ! Il reste 4 leprechauns.
Le leprechaun 8 disparait ! Il reste 3 leprechauns.
Le leprechaun 3 disparait ! Il reste 2 leprechauns.

Les 2 gagnants sont les leprechauns 1 et 4.
0.09923219680786133 secondes
```

Votre fichier `TP3.py` doit avoir une fonction `main(n)` qui prend le nombre de leprechauns en argument. Ainsi, on doit pouvoir appeler votre algorithme avec par exemple la commande `TP3.main(100)`. Remettez sur Studium **tous** les fichiers nécessaires à l'exécution.