TME2:

Nous avons calculé à la main le nombre de façon possible de disposer chacun des bateaux sur une grille de 10x10.

Bateau	Taille	nb
Porte avion	5	120
Croiseur	4	140
Contre-torpilleurs	3	160
Sous-marin	3	160
torpilleur	2	180

Ce qui nous donne alors comme nombre de configuration maximal théorique : 7.74×10^{10}.

Les fonctions qui traitent du combinatoire se trouvent dans le dossier `src/utils/comb.py`. L'utilisation du bout de code suivant nous permet de trouver la même valeurs que nous résultats théoriques.

```
nb_tot: int = 1
for i in range(1,len(LEN_B)+1):
    nb_tot *= nb_placer(i)
```

Cette valeur cependant inclue les tableaux où les bateaux peuvent se superposer ce qui n'est pas possible dans les règles du jeu. Pour avoir une valeur plus précise, nous devons ignorer tout ces plateaux.

La fonction `nb_placerL` fonction (*naïve toujours*) marche très bien pour des petites listes de bateaux ou pour des petites grilles, mais pour notre cas le temps d'excecution est énorme, même en optimisant avec les symètries cela prendrais toujours trop de temps, notre fonction se rapproche d'une complexité $O((3n)^m)$ avec n la taille de la dimension du plateau (supposé carré) et m le nombre de bateau.

Python n'est pas le langage pour ce genre de calcul intensif, on sait que notre résultat doit être aux alentours de 10^{10}. Le bout de code simplite suivant prend déjà trop longtemps à s'excécuter :

```
i = 0
while i < 1e10:
    i+=1
```

Essayons une autre approche, en supposons toutes les grilles équiprobable, on a donc : $P(\text{table}_n) = \frac{1}{N}$ avec $P(\text{table}_n)$ la chance de tirer une table aléatoire et N le nombre de table au totale.

Si on prend une fonction qui prend en paramètre une grille et génère des grilles aléatoirement jusqu'à ce que la grille générée soit la même que celle passé en

paramètre. Alors dans ce cas le nombre de grille généré sera une approximation de nombre totale de tableau