Exercices ★

I. Comparaison parfaite

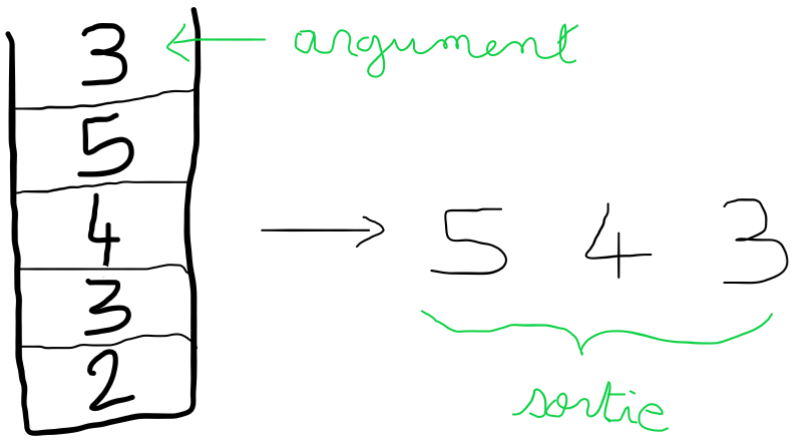
1. Implémenter une mauvaise mauvaise fonction qui prends en argument deux entiers x et y et renvoie le booléen exact $x > y$

L'idée peut être une version adaptée de ce qui a été fait pour l'égalité précédemment.

II. Qui sont-ils ?

2. Écrire une mauvaise mauvaise fonction qui prends en argument un entier n et affiche les n premiers éléments sur la pile (argument exclu), on supposera que ce sont tous des entiers.

Le but de cet exercice est d'afficher les premiers éléments de la pile, donc les éléments avec une profondeur minimale, en fonction d'un paramètre n, l'élément au dessus de la pile.
L'affichage peut être quelconque, mais les éléments doivent apparaître du moins profond au plus profond.
Attention, pour être lisible, on pourras penser à afficher des espaces (ou autre) pour séparer les éléments.
Voici un exemple de pile et de sortie attendu du programme (l'affichage peut varier mais le contenu doit être le même) :



III. Le retour de la somme

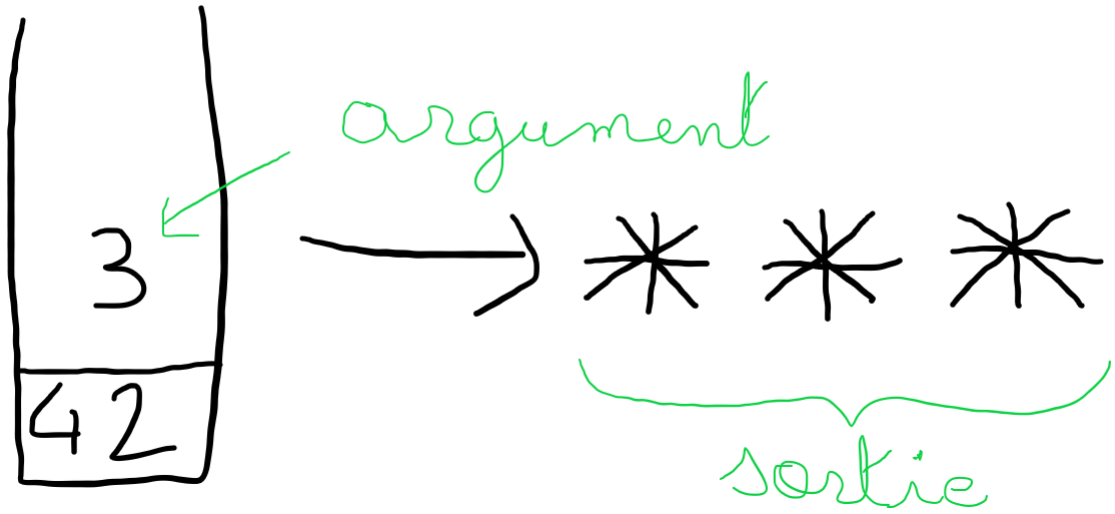
3. Faire une mauvaise mauvaise fonction qui prends en argument un entier et affiche la somme de 1 à cet entier.

La fameuse somme, le but est de calculer $\sum_{k=0}^{n} k$ avec n l'argument de notre fonction. Conseil, vous pouvez envisager de calculer la somme de n à 0 plutôt que de 0 à n, cela sera peut être plus simple, et pensez à utiliser ce que l'on a fait dans les questions précédentes.

IV. Plein d'étoiles

4. Faire une mauvaise mauvaise fonction qui prends en argument un entier et qui affiche une ligne d'autant d'étoiles (le caractère $*$).

Le but de cet exercice est d'afficher plein d'étoile ($*$), toutes en ligne, et en fonction d'un entier donné en argument. Voici la sortie attendu (en fonction de la pile à gauche) :



V. Factorielles

5. Implémenter une fonction qui traduit la fonction factorielle.

L'idée ici est simple, prendre en argument un entier n et d'afficher (ou de mettre au sommet de la pile peut importe) l'entier $n!$. Ici, cela peut être fait en faisant une sorte de «boucle», mais une approche récursive est beaucoup plus simple (croyez moi, même si en fait c'est similaire, penser récursivement est impératif[1] dans ce langage).

[1] La blague n'était pas voulue