❑ Exercice 1 : *Dessin récursif*

On souhaite écrire une fonction qui affiche à l'écran le dessin suivant :

```
*
**
***
****
```

Où le nombre de lignes affichées est le paramètre `n` de la fonction.

1. Ecrire en pseudo langage une version itérative de cette fonction.
2. Ecrire en pseudo langage une version récursive de cette fonction.

❑ Exercice 2 : *Exponentiation rapide version récursive*

1. Ecrire en pseudo langage l'algorithme d"exponentiation rapide récursif vu en cours.
2. Prouver que cet algorithme termine.
3. Prouver qu'il est totalement correct.

❑ Exercice 3 : *Somme des éléments d'un tableau*

1. Ecrire un algorithme récursif permettant de calculer la somme des éléments d'un tableau.
2. Prouver que cet algorithme est totalement correcte.

❑ Exercice 4 : *Calcul du* pgcd

1. Ecrire une version récursive de l'algorithme d'Euclide de calcul du pgcd.
2. Prouver que cet algorithme termine.
3. Prouver qu'il est totalement correcte.

❑ Exercice 5 : *Palindrome*

1. Ecrire une version récursive d'un algorithme permettant de vérifier qu'une chaine de caractère est un palindrome.
2. Prouver que cet algorithme termine.
3. Prouver qu'il est totalement correcte.

❑ Exercice 6 : *Coefficient du binôme*

1. Rappeler la relation de récurrence liant les coefficients du binôme $\binom{n}{k}$.
2. Donner les valeurs de $\binom{n}{0}$ et $\binom{n}{n}$.
3. En déduire un algorithme récursif permettant de calculer $\binom{n}{k}$.
4. Tracer l'arbre des appels récursifs pour le calcul de $\binom{4}{3}$. Que peut-on en déduire ?

❑ Exercice 7 : *Correction de l'exponentiation rapide itérative*

On rappelle l'algorithme d'exponentiation rapide en version itérative vu en cours :

Algorithme : Version itérative de l'exponentiation rapide

Entrées : $a \in \mathbb{R}^{*+}, n \in \mathbb{N}$
Sorties : a^n

$p \leftarrow 1$
tant que $n \neq 0$ **faire**
  si n *est impair* **alors**
    $p \leftarrow p \times a$
  fin
  $a \leftarrow a * a$
  $n \leftarrow \lfloor \frac{n}{2} \rfloor$
fin
return p

1. Prouver que cet algorithme termine.
2. Prouver que cet algorithme est totalement correct.
   On pourra prouver l'invariant suivant : $p \times a^n = a_0^{n_0}$ où a_0 (resp. n_0) désigne la valeur initiale de a (resp n).