Entraînement : concevoir une fonction récursive

Les solutions peuvent être rédigées en **pseudo-code, python, C, C++ ou Java**. La syntaxe du langage n'a pas d'importance tant que celle-ci reste **cohérente** et **compréhensible**. (Dans les exemples, les solutions sont données en pseudo-code).

Niveau 1

Grille d'évaluation

A (20)	L'algorithme est récursif et répond au problème donné. La complexité calculée est correcte (cohérente avec l'algorithme).
B (16)	La partie récursive de l'algorithme est correcte mais il y des erreurs sur les cas d'arrêts ou cas particuliers OU sur la complexité calculée.
C (11)	La partie récursive de l'algorithme est correcte mais il y a des erreurs sur les cas d'arrêts ou cas particuliers ET sur la complexité.
D (8)	Erreur sur la partie récursive de l'algorithme ou cas d'arrêt complètement absents.
E (1)	Algorithme non récursif ou récursif sans cas d'arrêts et faux.

Exercice 1.
Les coefficients binomiaux $\binom{n}{k}$ sont des nombres mathématiques qui donnent le nombre de façon de choisir k objets parmi n. Ils vérifient en particulier les propriétés suivantes :

- $\binom{n}{n} = \binom{n}{0} = 1$,
- si $k < 0$ ou $k > n$, $\binom{n}{k} = 0$,
- Pour $n > 0$, $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

(1) Donner un algorithme **récursif** qui calcule le coefficient binomial $\binom{n}{k}$. Votre fonction devra retourner un résultat (et donc terminer) quels que soient les entiers n et k passés en paramètre.

(2) Quelle est la complexité de votre algorithme ?

Solution

```
Binomimal
Input : deux entiers n et k
Procédé :
    Si n = k ou k = 0 :
        Retourner 1
    Si k < 0 ou k > n:
        Retourner 0
    Retourner Binomal(n-1,k-1) + Binomial(n-1, k)
```

Complexité $O(2^n)$.

Niveau 2 (bonus en partiel)

Grille d'évaluation

A (20)	L'algorithme est récursif et répond au problème donné.
B (16)	Le principe récursif est en partie correct mais il y a des erreurs dans son implantation et / ou dans les cas particuliers et conditions d'arrêt.
C (11)	Un algorithme récursif est proposé qui tente de répondre à la question posée même s'il ne répond pas aux spécifications du problème.

Exercice 2.
Écrire une fonction qui prend en paramètre une chaîne de caractère ainsi qu'un entier k et retourne la liste de toutes les façon de choisir k lettres dans la chaîne.

Par exemple, si je lance ma fonction sur `"chat"` et 2, elle devra renvoyer : `[ "ch", "ca", "ct", "ha", "ht", "at"]`. Remarquez que l'ordre des des lettres reste le même que dans la chaîne originale. L'ordre des mots dans la liste n'a pas d'importance. La fonction ne s'occupera pas de supprimer les éventuels doublons et on retournera donc toujours une liste de taille $\binom{n}{k}$ où n est la taille de la chaîne.

On pourra utiliser la syntaxe python pour les listes et les chaînes de caractères :
Liste vide : `L <- [ ]`
Ajout : `L.append(valeur)`
Première lettre d'un mot : `u[0]`
Dernière lettre d'un mot : `u[-1]`
Concaténation de deux chaînes de caractère : `u + v`
La chaîne de caractère obtenue en copiant les lettres de `u` à partir de l'indice 1 : `u[1 :]`
La chaîne de caractère obtenue en copiant les lettres de `u` sauf la dernière : `u[ :-1]`

Solution

```
KParmiN
Input : une chaine de caractère s et un entier k
Procédé :
    Si k = 0:
        Retourner [""] # liste contenant le mot vide
    Si s = "":
        Retourner [] # liste vide (pas de résultat)
    resultat = []
    Pour e dans KParmiN(s[1:], k-1):
        resultat.append(s[0] + e)
    Pour e dans KParmiN(s[1:], k):
        resultat.append(e)
    Retourner resultat
```