# Devoir surveillé

**Matière** : Algorithmique et structures de données I
**Enseignants** : Majdi Jribi et Khaoula Bezzina
**Filière** : MPI
**Nombre de pages** : 2

**Semestre:** 1
**Date:** 01 Décembre 2020
**Durée:** 1h30mn
**Documents :** ***non autorisés***

**Chaque exercice doit être rédigé sur une feuille d'examen à part.**

**L'utilisation du type Tableau est strictement interdite.**

**L'utilisation du type chaine de caractères est strictement interdite.**

## Exercice 1 *(5 pts)*

Un **entier positif K** à **n** chiffres est dit nombre de **KAPREKAR** si la somme du nombre composé de **n** chiffres de droite à gauche de $K^2$ avec le nombre composé par le reste des chiffres de $K^2$ est égale à **K.**

**Exemples** : 9, 45 et 297 sont des nombres **KAPREKAR**

Pour **9**, n=1, $9^2 = 81$. Le nombre avec les n chiffres de droite à gauche est **1**.
Le nombre composé par les chiffres qui restent est **8.** 1+8=9 qui est égal au nombre de départ 9.

Pour 45, n=2, $45^2 = 2025$. Le nombre avec les n chiffres de droite à gauche est **25**.
Le nombre composé par les chiffres qui restent est **20.** 25+20=45 qui est égal au nombre de départ 45.

1- Ecrire un algorithme qui permet de lire un entier positif et qui permet de vérifier s'il est un nombre de **KAPREKAR.**

## Exercice 2 *(8 pts)*

La suite de Conway est une suite de chiffres où chaque terme est l'énonciation des nombres de chiffres identiques successifs du terme précédent.
Pour générer un terme de la suite, utiliser le précédent en le lisant chiffre après chiffre et en regroupant les chiffres qui se répètent consécutivement. Une séquence du même chiffre est remplacée par le nombre de répétition suivi par le chiffre répété.

**Par exemple : 111** est remplacé par 31. 3 est le nombre de répétition de 1 successivement et 1 est le chiffre qui se répète.

**Le premier terme de la suite est 1**

Exemple :

| Indice du terme | Terme | Se lit | S'écrit (Terme suivant) |
|---|---|---|---|
| 1 | 1 | Un 1 | 11 |
| 2 | 11 | Deux 1 | 21 |
| 3 | 21 | Un 2 et un 1 | 1211 |
| 4 | 1211 | Un 1, un 2 et deux 1 | 111221 |
| 5 | 111221 | Trois 1, deux 2 et un 1 | 312211 |

La suite est donc 1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, etc.

1- Ecrire en algorithmique puis en langage C la fonction ***CalculTaille()*** qui permet de retourner la taille (nombre de chiffres) d'un entier n strictement positif.

2- Ecrire en algorithmique puis en langage C la fonction ***ConcatenerEntiers()*** qui permet de retourner la concaténation de deux entiers strictement positifs. On suppose que le type du résultat est **entier.**

**Exemple** si x=123 et y=256 alors le résultat de la concaténation est 123256

3- Ecrire en algorithmique puis en langage C la fonction ***TermeSuivant()*** qui pour un terme donné $t_i$ strictement positif, renvoie le terme suivant $t_{i+1}$.

**Exemple** : si $t_i$=312211, $t_{i+1}$=13112221

## Exercice 3 *(7 pts)*

Soient a, b et n trois entiers strictement supérieurs à zéro. On dit que a et b sont factoriellement égaux pour l'ordre n s'ils vérifient :

$$\frac{(n+1)*b}{a!} = \frac{a}{n*b!}$$

Avec (**m !**) est la factorielle de l'entier m.
Nous rappelons que : **m !=1*2*..*(m-1)*m** pour m>0.

1- Ecrire en algorithmique la fonction ou la procédure ***Lecture_3entier()*** qui permet de lire trois entiers a, b et n strictement supérieurs à zéro à partir du clavier.

2- Ecrire en algorithmique la fonction ou la procédure ***Verification_fact()*** qui, pour trois entiers a, b et n strictement supérieurs à zéro, vérifie si a et b sont factoriellement égaux pour l'ordre n.

**Remarque importante :** Pour une valeur entière ***m*** très grande, il est impossible de trouver un type de données qui pourra supporter la valeur de (***m!***). La réponse à la question 2 doit éviter le calcul direct de (***a!***) et (***b!***).

3- Ecrire un algorithme qui permet de lire deux entiers ***a*** et ***b*** strictement supérieurs à zéro à partir du clavier et qui détermine le **premier** entier **n** strictement positif pour lequel a et b sont factoriellement égaux pour l'ordre **n**.

4- Traduire en langage C la question 1 et la question 3