TP N°4: Conception d'algorithmes élémentaires et manipulation de tableaux.

Durée : 3 heures

Le but de cette séance est d'améliorer encore la partie conception d'algorithmes ainsi que l'utilisation de méthodes. Les tableaux seront revus.

Exercice 1 : Rappel sur les méthodes **(1 point)**

Ecrire un programme qui calcule et affiche la factorielle d'un nombre tapé au clavier. La méthode factorielle sera écrite en dehors de la méthode main, mais sera appelée depuis l'intérieur de la méthode main.

Exercice 2 : Le triangle de Pascal (3 points)

Le triangle de Pascal est composé de M + 1 lignes consécutives donnant toutes les valeurs des Cnp pour n variant de 0 `a M et p variant de 0 à n :
On rappelle que :

$$C_n^p = \frac{n!}{p!(n-p)!}$$

Ecrire un programme qui calcule et affiche les coefficients du triangle de Pascal pour une valeur M donnée. Pour M = 5, le résultat est le suivant :

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
```

Exercice 3 : Plus petite valeur des éléments d'un tableau (3 points)

Ecrire un programme qui détermine et affiche la plus petite valeur des éléments d'un tableau. On complétera le programme suivant :

```
public class Valeurmax {
    public static void main(String[] args) {
    // Définition d'un tableau de 10 entiers
    int [] Tableau = {5, 3, 12, 4, 7, 9, 1, 8, 19, 2};

    }
}
```

Afin de gagner du temps sur l'exercice suivant, on écrira une méthode qui prend en paramètre le tableau et qui renvoie la valeur du plus petit ainsi que son index. Une méthode ne pouvant renvoyer qu'un seul objet, elle renverra un tableau de 2 valeur : le plus petit et l'index .

Exercice 4 : Tri par sélection (3 points)

Il s'agit d''écrire un programme réalisant le tri de données. La méthode de tri utilisée est basée sur la sélection du minimum. Pour chaque indice i du tableau, on recherche à partir de cet indice le plus petit élément du tableau et on permute cet élément avec celui se trouvant à l'indice i. Affichez le tableau à chaque itération.

Aide : Pour afficher un tableau, vous pouvez importer le package Arrays qui se trouve dans java.util : (*import java.util.Arrays;*). Ainsi l'affichage d'un tableau s'écrit simplement :

System.out.println(Arrays.toString(Tableau)) ;

Exercice 5 : Votre générateur de séquence aléatoire personnel (3 points)

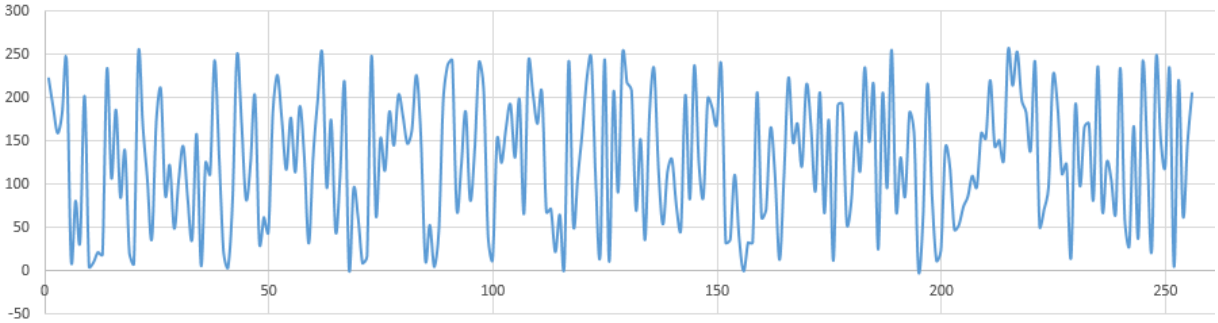
Ecrivez le code de la méthode « ***suivant*** » qui correspond à l'algorithme que vous aviez préparé en tutorat. (C'est important que vous utilisiez le vôtre, même si les séquences obtenues sont courtes)
Ecrivez l'algorithme ci-dessous afin de remplir un tableau de 256 cases puis affichez le tableau obtenu dans la console. (Normalement vu en tutorat)

```
1.  ALGORITHME Tirage

2.      ENTIER : tableau[256]
3.      ENTIER : germe,i

4.   DEBUT
        Tableau[0]<- germe
5.      POUR i ALLANT_DE 1 A 255 PAR PAS DE 1
           Tableau[i]<-SUIVANT(Tableau[i-1])
6.
7.      FIN_POUR
8.   FIN
```

En fonction des résultats obtenus modifiez les paramètres de votre générateur de telle manière à ce que la suite ne converge pas et produise une séquence de longueur supérieure à 20. De manière générale, si vous faites n'importe quel calcul avec des nombres premiers et que vous appliquez un modulo 256 cela devrait fonctionner après quelques essais. Aussi, il est inutile de chercher un sequence « parfaite » à la main, nous laisserons l'ordinateur le faire pour nous en fin de TP.
Dans un premier temps, vous pouvez copier-coller les valeurs depuis la console dans Excel ou Matlab pour visualiser graphiquement la séquence obtenue :



Exemple de visualisation sous excel

Ecrire la méthode qui calcule et affiche la période de la séquence obtenue.

Exercice 6 : Zéro-Crossing (3 points)

Pour vérifier l'aspect aléatoire de la séquence produite, on se propose de calculer la valeur moyenne de la séquence et de la soustraire à chaque valeur du tableau. La séquence devient donc à moyenne nulle.

Ecrivez une méthode qui prend en paramètre le tableau, qui calcul et soustrait la valeur moyenne puis qui compte combien de fois le « signal » obtenu croise l'axe des abscisses. (ou autrement dit passe par zéro, ou autrement dit que le signe change entre 2 valeurs consécutives).
Plus ce croisement aura lieu souvent, plus le signal sera considéré comme dynamique et donc aléatoire.
On considèrera par la suite que la séquence est valide si le nombre de passage par zéro est compris entre 100 et 150.

Exercice 7: Optimisation selon un critère (4 points)

De manière automatisée (avec des boucles for), faire un programme qui recherche les paramètres optimaux de votre générateur (constantes multiplicatives, additives, puissances etc) en utilisant comme critère la longueur de la période générée et en imposant comme contrainte un zéro crossing compris entre 100 et 150. L'enseignant pourra vous guider sur les paramètres à optimiser en fonction du générateur que vous avez imaginé. L'idée est

d’essayer toutes les constantes possibles sur une plage donnée et de retenir automatiquement celles qui fonctionnent le mieux.