TP 14 Java

Exercice 1 Utiliser le code ci-dessous pour écrire un fichier `Exo1.java` qui permet d'afficher la valeur minimale du tableau `tab`. Il faut écrire que la fonction principale `main`.

```
class ....{
    public ... void main (String[]...){
        int[] tab = {3, 9, 23,1,29,134,12,667,43,2,120,32,123};
        int min = tab[0];
        int apparitions_min=1;
        for(int i ; i<tab.length; ...)
            if(min==tab[i]){
                apparitions_min ++; //une apparition de plus de la valeur min
            }
            if(.....){
                min=tab[i];
                apparitions_min = 1; //1ère apparition de la nouvelle valeur min
            }
    }
}
```

Exercice 2 Continuer le programme précédent dans un nouveau fichier `Exo2.java`. L'objectif est de calculer `apparitions_min` d'une autre manière.
Écrire d'abord une fonction `int nbVals(int[] tab, int valeur)` qui renvoie le nombre d'apparitions de la valeur `valeur` dans le tableau `tab`. Utiliser cette fonction pour afficher le nombre d'apparitions de la valeur `min` calculée dans le `main`.

Exercice 3 Écrire une classe `Exo3` avec une fonction une fonction `i_puissance_i(int i)` qui renvoie i^i à l'aide d'une boucle `for`.

Exercice 4 Écrire une classe `Exo4` avec une fonction `sommePuissances(int n)` qui renvoie la somme $1^1+2^2+3^3+\ldots n^n$. Par exemple, pour $n = 5$, il faut renvoyer $1^2+2^2+3^3+4^4+5^5 = 1+4+27+256+3125 = 3413$. N'hésitez pas à utiliser la fonction `i_puissance_i(int i)` de l'exercice précédent pour écrire la nouvelle fonction.

Exercice 5 Écrire une classe `Exo5` avec une fonction `puissance_parfaite(int x)` qui renvoie `true` si `x` peut s'écrire sous la forme $\texttt{x} = \texttt{i}^{\texttt{i}}$ (ex, $256 = 4^4$, $3125 = 5^5$) ou `false` sinon.

Astuce Vous pouvez faire une boucle `for` pour parcourir les valeurs $i = 1, 2, 3 \ldots,$ `x`. S'il y a un seul i tel que $i^i = x$ (utilisez la fonction `i_puissance_i(...)` pour vérifier cela), alors il faut renvoyer `true`.

Exercice 6 Écrire un programme `Exo6.java` et une fonction `int compterPuissancesParfaites(int[] t)` qui calcule le nombre de puissances parfaites dans le tableau `t`. Cette fonction doit faire appel à la fonction `puissance_parfaite` de l'exo précédent. Tester la fonction dans le `main` sur un tableau comme :

```
int[] tab = {8, 1, 27, 9,-1,2,7,4,3125};
```

Exercice 7 Écrire un programme `Exo7.java` qui demande à l'utilisateur de saisir 5 valeurs ; vous pouvez vous inspirez du code ci-dessous. Pour chaque saisie, il faut indiquer à l'utilisateur s'il a saisit une puissance parfaite ou pas.

```
double[] vals = new double[5];
Scanner scan = new Scanner(System.in);
System.out.println("Entrer 5 valeurs");
for(int i=0;i<5;i++)
    vals[i] = scan.nextDouble();
```

Exercice 8 Écrire une fonction statique d'en-tête

```
int trouveMinMat(int[][] m)
```

qui renvoie la valeur minimale d'une matrice carré `m`. Tester cette méthode sur une matrice comme par exemple :

```
public static void main(...){
    int[][] matrice = { {2, 8, 9},
                        {90,2, 70},
                        {0, 1, 3}};
    ..println(trouveMinMat(...));
}
```

Vous pouvez vous inspirer du code ci-après :

```
}
static int ...                    {
    ... n = m.length;
    min = ..
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++)
            if(m[i][j] ... min){
                min =
            }
    }
}
```

Exercice 9 Écrire un programme `Exo9.java` avec une fonction `double somme(n)` qui renvoie la somme $1+2+\cdots+n$. Continuer le programme et ajouter une deuxième fonction qui renvoie :

$$\frac{1}{\texttt{somme(1)}}+\frac{1}{\texttt{somme(2)}}+\frac{1}{\texttt{somme(3)}}+\ldots\cdots+\frac{1}{\texttt{somme(n)}}$$

Regardez si la valeur renvoyée par cette fonction se raproche de `2` lorsque n devient très grand.

Exercice 10 Écrire un programme `Exo10.java` avec une fonction `double fact(n)` qui calcule le produit $1\cdot 2\cdot\cdots\cdot n$. Si $n\leq 0$, la fonction doit renvoyer 1. Continuer le programme et ajouter une deuxième fonction qui renvoie :

$$\frac{1}{\texttt{fact(0)}}+\frac{1}{\texttt{fact(1)}}+\frac{1}{\texttt{fact(2)}}+\frac{1}{\texttt{fact(3)}}+\ldots\cdots+\frac{1}{\texttt{fact(n)}}$$

Regardez si la valeur renvoyée par cette fonction se raproche de `Math.E` lorsque n devient très grand.

Exercice 11 Écrire un programme `Exo11.java` avec une fonction `double racine(`x,n`)` qui servira à calculer la racine carré de x. On démarre avec $x_i = x$ et on applique n fois la formule $x_{i+1} = \dfrac{x_i + x/x_i}{2}$. Pour $n \geq 10$, x_n sera une bonne approximation de $\sqrt{x}$.

Exercice 12 Modifiez la fonction précédente ainsi : on n'utilise plus d'argument `n`, mais on arrête les itérations dès que x_{i+1} devient trop proche de x_i. Par trop proche, on comprends que la valeur absolue $|x_{i+1} - x_i|$ est inférieure à 0.0000001 !