4TPM205U: Algorithmique des tableaux: feuille 2

Itérer avec la boucle `while`

Travaux dirigés

Exercice 1 – ***(Division euclidienne)***

Soient a un entier positif et b un entier strictement positif. Effectuer la division euclidienne (division entière) de a par b, c'est déterminer l'unique couple d'entiers (q, r) tels que $a = bq + r$ et $0 \leq r < b$.

1. Écrire un algorithme de division euclidienne par soustractions successives. Faire tourner l'algorithme pour $a = 13$ et $b = 3$; pour $a = 2$ et $b = 7$.
2. Déterminer, en fonction de a et b, le nombre d'opérations élémentaires effectuées (comparaisons, additions, soustractions).
3. Écrire deux fonctions `reste(a,b)` et `quotient(a,b)` qui renvoient respectivement le reste et le quotient de la division entière de `a` par `b`.

Exercice 2

On rappelle qu'en base 10 :

– le ***reste*** de la division de n par 10 est égal à la valeur du dernier chiffre de n (celui situé le plus à droite) ;

– le ***quotient*** de la division entière de n par 10 est égal à la valeur représentée par le nombre n privé de son dernier chiffre.

Écrire une fonction `sommeChiffres` qui renvoie la somme des chiffres décimaux d'un nombre entier n passé en paramètre.

Exercice 3 – Soit la fonction suivante :

```
def mystere (n, x):
    while (n // x > x and x != 0) :
        n = n // 10
        x = x - 1
    return n
```

Quelle est la valeur renvoyée par `mystere(100,6)` ? et par `mystere(1000,3)` ?

Corriger la condition d'itération afin qu'elle ne produise jamais d'erreur.

Exercice 4 – *(Lancers de dés à 6 faces)*

On suppose qu'on dispose d'une fonction `lancerDe()` qui simule le lancer d'un dé à 6 faces : l'appel de cette fonction renvoie un entier entre 1 et 6 choisi de manière aléatoire uniforme.

1. Écrivez une fonction `nbLancers6()` qui renvoie le nombre de lancers d'un dé nécessaires pour obtenir un 6. Si le dé sort 1, puis 5, 2, et 6, la fonction renverra la valeur 4.
2. On lance maintenant deux dés à la fois. Écrivez une fonction `nbLancersDouble()` qui renvoie le nombre de lancers nécessaires pour obtenir un double (*i.e.*, lorsque les deux dés ont la même valeur).
3. Écrivez une fonction `moyenneTentativesDouble(n)` qui calcule la moyenne du nombre de tentatives faites pour obtenir un double, prise sur un échantillon de `n` doubles obtenus. Par exemple si `n` vaut 10, on lancera les dés pour obtenir 10 fois des doubles et on calculera la moyenne du nombre de tentatives sur ces 10 séquences.
4. Écrivez une fonction `nbLancersPourSomme(n, valeur)` qui calcule le nombre de lancers faits avec `n` dés pour obtenir une somme des dés égale à `valeur`. Que faire si la `valeur` passée en paramètre de la fonction n'est pas dans le domaine de définition des valeurs possibles des `n` dés ?

Exercice 5 – *(Racine carrée entière)*

La somme des n premiers nombres impairs est égale au carré de n.

Par exemple, $1+3+5=3^2$. Autre exemple, $1+3+5+7=4^2$, etc.

En déduire un algorithme pour calculer la racine carrée entière par défaut d'un nombre entier positif donné ; calculer le nombre d'additions et de comparaisons à effectuer.