Examen du jeudi 9 novembre 2023

durée : 2 heures

Pour chacun des algorithmes, on justifiera avec soin :
- que l'algorithme termine (si une boucle « tant que » est utilisée).
- que l'algorithme renvoie bien le résultat demandé.

Une réponse non justifiée sera notée sur les trois quarts des points.

Pour les boucles « pour », on adoptera les conventions suivantes. Si $a, b \in \mathbb{Z}$, « pour i allant de a à b » signifie « pour i parcourant en croissant l'intervalle $[\![a, b]\!]$ ». Lorsque $b < a$, cet intervalle est vide, donc aucune des instructions dans la boucle « pour » n'est effectuée (ce sera par exemple le cas si on écrit « pour i allant de 1 à n », avec $n = 0$). On pourra utiliser l'instruction « pour i allant en décroissant de b à a », qui signifie (si $b \geq a$) « pour i prenant successivement les valeurs b, $b-1$,..., a ». Lorsque $a > b$, aucune instruction dans la boucle « pour » n'est alors effectuée.

Exercice 1. *Écrire un algorithme pour calculer chacune des valeurs suivantes :*

1. *pour $x \in \mathbb{R}$ donné, calculer $|x|$,*
2. *pour $n \in \mathbb{N}$ donné, calculer $n!$,*
3. *pour $x \in \mathbb{R}$ et $n \in \mathbb{N}$ donnés, calculer x^n, de manière itérative, puis récursive, (de manière basique, sans exponentiation rapide),*
4. *pour $x \in \mathbb{R}$ et $n \in \mathbb{N}$ donnés, calculer $S = \sum_{k=0}^{n} \frac{x^k}{k!}$; pour ce calcul, on donnera un algorithme basique utilisant les algorithmes précédents, et un algorithme minimisant le nombre d'opérations,*
5. *le maximum d'une liste L non-vide.*

Exercice 2.
1. *Soit (u_n) la suite définie comme suit : $u_0 = 1$, $u_1 = 2$ et $u_{n+2} = 3u_n + 2u_{n+1}$. Écrire un algorithme itératif qui prend en entrée élément $n \in \mathbb{N}$ et qui renvoie u_n.*
2. *Écrire un algorithme récursif qui prend en entrée élément $n \in \mathbb{N}$ et qui renvoie u_n.*
3. *Soit $(v_n) \in \mathbb{R}^{\mathbb{N}^*}$ la suite définie comme suit : $v_1 = 1$, $v_2 = 3$ et si $n \in \mathbb{N}$, $\begin{cases} v_{2n} = v_n^2 + 5 \\ v_{2n+1} = v_n v_{n+1} + 7 \end{cases}$.*
   Écrire un algorithme qui prend en entrée un élément $n \in \mathbb{N}^*$ et qui détermine v_n.

Exercice 3. *Soit $L = [a_0, \ldots, a_{n-1}]$, où les a_i sont des entiers. On rappelle le principe du tri par sélection, pour trier la liste L : on cherche d'abord l'entier (ou un des entiers) i tel que a_i est le plus petit élément de la liste. On échange ensuite a_0 et a_i. On obtient alors une suite $L_1 = [a_i, b_1, , \ldots, b_{n-1}]$. On réitère ensuite le processus avec $[b_1, \ldots, b_{n-1}]$, et ainsi de suite.*

Écrire un algorithme qui prend en entrée une liste d'entiers et qui la trie, en utilisant le tri par sélection.

Exercice 4. *(équation de Pell-Fermat) On considère l'équation (E) $x^2 - 2y^2 = 1$, d'inconnues $x, y \in \mathbb{N}^*$. Écrire un algorithme qui renvoie la liste de tous les couples (x, y) qui sont solution de (E) et qui vérifient $y \leq 100$.*

Exercice 5.
1. *Pour $n \in \mathbb{N}^*$, on pose $u_n = \sum_{k=0}^{n} \frac{1}{k!}$ et $v_n = u_n + \frac{1}{n!.n}$. Montrer que (u_n) et (v_n) sont adjacentes.*
2. *On admet que (u_n) tend vers e. Écrire un algorithme qui prend en entrée en réel positif ϵ et qui renvoie une valeur approchée à ϵ près de e.*

Exercice 6. 1. *On rappelle que si $a, b \in \mathbb{N}^*$, alors $a \wedge b = b \wedge r$, où r est le reste dans la division euclidienne de a par b. On propose l'algorithme suivant, qui prend en entrée deux entiers $a, b \in \mathbb{N}$ et qui est censé déterminer leur PGCD. Fonctionne-t-il ? Si oui, le justifier, sinon, le corriger.*

***Algorithme**(a, b) :*

```
si a <= b :
    | b, a ← a, b
Tant que b ≠ 0 :
    | b ← a
    |b← a%b
Renvoyer a.
```

2. *Une puce se déplace sur un axe gradué que l'on identifie avec $\mathbb{Z}$. Au temps $t = 0$, la puce est en 0. Supposons que la puce est en x à l'instant n. Alors à l'instant $n + 1$, elle est en $x + 1$ avec probabilité $1/3$, en $x + 2$ avec probabilité $1/3$, en $x - 3$ avec probabilité $1/3$. On souhaite programmer un algorithme simulant la position de la puce après 50 itérations. On propose l'algorithme suivant :*

***Algorithme** :*

```
x = 0
Pour t allant de 1 à 50 faire :
    | Si random()< 1/3 :
        | x ← x + 1
    | Si random()>= 1/3 et random()< 2/3 :
        | x ← x + 2
    |Si random()> 2/3
        | x ← x − 3
Renvoyer x.
```

Cet algorithme fonctionne-t-il ? Si oui, le justifier, sinon, le corriger (la fonction random() retourne un nombre (pseudo)-aléatoire entre 0 et 1).