Corrigé

AL 5 ppa - Algorithmique avancée - contrôle continu

Nom : _____

Prénom : _____

- Durée : 1 heure 30. Documents interdits, calculatrices interdites.
- Écrivez toutes les réponses directement sur le sujet. Si vous n'avez pas suffisamment de place, écrivez au dos d'une feuille en le **précisant dans la question**.

1 Tests de primarité (9 points)

Le but de cette section est l'élaboration d'un algorithme de test de primarité. Vous pourrez utiliser les fonctions suivantes :

- $divise(x, y)$ retourne vrai si et seulement si x divise y.
- $modulo(x, y)$ retourne le reste de la division entière de x par y, la valeur retournée se trouve toujours entre 0 et $y - 1$.
- $modExp(base, exp, m)$ retourne $base^{exp}\ mod\ m$
- $random()$ retourne un nombre entier aléatoire.

1. [2 points] Ecrire en pseudo-code avec les conventions de votre choix une fonction non récursive $estPremier$ prenant un entier a en paramètre et retournant $vrai$ si et seulement si a est un nombre premier. Vous testerez la primarité de a en le divisant successivement par tous les entiers compris entre 2 et $\sqrt{a}$.

Solution:

```
Fonction estPremier(a)
    i ⟵ 2
    tant que i² ≤ a
        si divise(i, a) alors
            retourner vrai
        fin
        i ⟵ i + 1
    fin tant que
    retourner faux
FIN
```

2. [2 points] L'algorithme précédent est d'une inefficacité avérée. Nous allons définir par la suite un algorithme davantage efficace. Nous avons besoin pour ce faire d'une fonction calculant le *pgcd*. Ecrire en pseudo-code avec les conventions de votre choix une fonction récursive $pgcd(x, y)$ retournant le *pgcd* de x et de y.

Solution:

```
Fonction pgcd(x, y)
    si y = 0 alors
        retourner x
    sinon
        retourner pgcd(y, modulo(x, y))
    fin
    retourner faux
FIN
```

3. [2 points] Nous allons utiliser le petit théorème de Fermat (si n est premier, alors pour tout a tel que $pgcd(a, n) = 1$, on a $a^{n-1} \ mod \ n = 1$). Il existe des couples (a, n) où n n'est pas premier pour lesquels on a quand même $a^{n-1} \ mod \ n = 1$, mais ces couples sont très rares. Cela signifie que si avec plusieurs a différents on a $a^{n-1} \ mod \ n = 1$, on a de très fortes chances que n soit premier. Ecrire une fonction $verifieFermat(a, n)$ retournant vrai si et seulement si $a^{n-1} \ mod \ n = 1$ (sans tester si $pgcd(a, n) = 1$).

Solution:

```
Fonction verifieFermat(a, n)
    retourner modExp(a, n − 1, n) = 1
FIN
```

4. [3 points] Ecrire une fonction $estPremier(n, nbA)$ testant avec nbA valeurs de a sélectionnées aléatoirement si n vérifie le petit théorème de Fermat. Pensez au fait que si $pgcd(a, n) \neq 1$ c'est que soit a est un multiple de n (ce qui ne nous donne aucune information sur la primarité de n), soit $pgcd(a, n)$ est un diviseur de n différent de 1 et de n.

Solution:

```
Fonction estPremier(n, nbA)
    i ⟵ 1
    tant que i ≤ nbA
        a ⟵ random()
        p ⟵ pgcd(a, n)
        si p = 1 alors
            si ¬verifieFermat(a, n) alors
                retourner faux
            fin
            i ⟵ i + 1
        sinon
            si divise(p, n) alors
                retourner faux
            fin
        fin
    fin tant que
    retourner vrai
FIN
```

2 Décomposition en produit de facteurs premiers (11 points)

Nous rappelons que tout entier naturel x peut se mettre de façon unique sous la forme $x = p_1^{e_1} p_2^{e_2} \dots p_n^{e_n}$ où p_i est le i-ème nombre premier. Ainsi un entier naturel x peut être représenté indifféremment en base 10 ou sous la forme d'un vecteur contenant les exposants $(e_1, e_2, \dots, e_n)$. Le but des exercices suivants est de mettre au point des fonctions de conversions entre ces deux représentations. Vous pourrez utiliser les fonctions suivantes :

- $p(i)$ retourne le i-ème nombre premier
- $puiss(base, exp)$ retourne $base^{exp}$
- $divise(x, y)$ retourne vrai si et seulement si x divise y.

1. 2 points Ecrire une fonction itérative $toBase10(e)$ retournant la représentation en base 10 du nombre dont la suite des exposants se trouve dans le tableau e à n éléments.

Solution:

```
Fonction toBase10(e)
    x ⟵ 1
    pour i ∈ {1, ..., n}
        x ⟵ x × puiss(p(i), e_i)
    fin pour
    retourner x
FIN
```

2. 3 points Ecrire la procédure $decompose(x, e)$ plaçant dans le tableau t les exposants $e_1, \dots, e_n$ de la décomposition en produit de facteurs premiers de x. Le tableau est supposé suffisamment grand et initialisé à 0.

Solution:

```
Procédure decompose(x, e)
    i ⟵ 1
    tant que x ≠ 1
        si divise(p(i), x) alors
            e_i ⟵ e_i + 1
            x ⟵ x/p(i)
        sinon
            i ⟵ i + 1
        fin
    fin tant que
FIN
```

3. [1 point] On rappelle que le nombre de diviseurs de $(e_1, \ldots, e_n)$ est $(e_1 + 1) \times (e_2 + 1) \times \ldots \times (e_n + 1)$. Ecrire la fonction non récursive $nbDiviseurs(e)$ retournant le nombre de diviseurs du nombre dont e contient la décomposition en produit de facteurs premiers.

Solution:

```
Fonction nbDiviseurs(e)
    nbd ⟵ 1
    pour i ∈ {1, ..., n}
        nbd ⟵ nbd × (e_i + 1)
    fin pour
    retourner nbd
FIN
```

4. [5 points] Nous utiliserons pour cet exercice des listes, vous vous servirez des fonctions suivantes :
   - $empty()$ retourne une liste ne contenant aucun élément.
   - $push(l, x)$ ajoute un élément x dans la liste l.
   - $mult(l, k)$ retourne une liste contenant tous les éléments de l multipliés par le nombre k.
   - $union(l, m)$ retourne une liste contenant tous les éléments de la liste l ainsi que tous les éléments de la liste m.

   Ecrire une fonction récursive $listeDiviseurs(e, i)$ retournant la liste des diviseurs du nombre dont la décomposition en produit de facteurs premiers se trouve dans e, tous les éléments se trouvant strictement avant l'indice i sont ignorés. Lors du premier appel à cette fonction, on a $i = 1$.

Solution:

```
Fonction listeDiviseurs(e, i)
    si i > n alors
        retourner push(empty(), 1)
    sinon
        l ⟵ listeDiviseurs(e, i + 1)
        m ⟵ empty()
        exp ⟵ 1
        tant que exp ≤ e_i
            m ⟵ union(m, mult(l, puiss(p(i), exp)))
            exp ⟵ exp + 1
        fin tant que
        retourner union(l, m)
    fin
FIN
```