Université de Bourgogne
Département de Mathématiques

L3 Mathématiques
Techniques de programmation

Contrôle terminal du 15 mai

Durée de l'épreuve : 2h

Vous devez choisir l'un des deux problèmes, chacun d'eux valant 20 points, et l'indiquer clairement sur votre feuille d'examen. Si vous travaillez sur les deux problèmes, votre note sera le score plus élevé et non la somme des scores des deux problèmes.

Chaque étudiant devra rendre obligatoirement une copie papier (même blanche) **et** un fichier Python ou Jupyter. Concernant le fichier informatique, il devra être sous le format "**NOM_PRENOM.py**" ou "**NOM_PRENOM.ipynb**". Puis il devra être déposé sur votre espace Plubel dans la section **Important : Examen**. Pour le dépôt, il suffit de cliquer sur **Contrôle terminal** puis **Ajouter un travail**. Vous pourrez ainsi ajouter votre document et **enregistrer**.

1. **Polynômes creux à coefficients réels**
   Définir une classe qui permette de manipuler des polynômes en utilisant notation creuse. Chaque polynôme peut être représenté par un tableau $2 \times N$, où N est le nombre de coefficients non nuls avec la première coordonnée qui indique les puissances et la seconde les coefficients associés ; par exemple $x^{101} + 2x^3$ peut être représenté par $[(101, 3), (1, 2)]$.

   (a) Implémenter des méthodes pour évaluer la somme, la soustraction, la multiplication, le reste de la division euclidienne, le quotient de la division euclidienne, et le PGCD de deux polynômes.

   (b) Trouver le quotient q et le reste r de la division euclidienne de A par B, $A = Bq + r$ oú
   $$A = x^{101} - 2, \quad B = x + 1.$$

   (c) Trouver le PGCD de P et Q oú
   $$P = 2x^{58} + 7x^{45} + 4x^{34} + 6x^{27} + 3x^{21} + 12x^{10}, \quad Q = x^{56} + 9x^{32} + 9x^8.$$

2. **Arbres binaires et K-aires**
   L'arbre binaire est défini comme une structure de données arborescente où chaque nœud a au plus 2 enfants. Étant donné que chaque élément d'un arbre binaire ne peut avoir que 2 enfants, nous les nommons généralement les enfants gauche et droit. Un arbre K-aire de recherche est une généralisation de notion de arbres binaires, où les nœuds peuvent avoir plus de deux enfants.

(a) Étant donné l'arbre binaire donné dans la figure 1 et les deux nœuds a et b, déterminez si les deux nœuds sont cousins l'un de l'autre ou non. Deux nœuds sont cousins l'un de l'autre s'ils sont au même niveau et ont des parents différents. Écrire un code qui trouve tous les cousins dans un arbre binaire arbitraire.

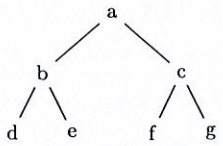

Figure 1: Les éléments d, e, f et g sont cousins, mais b et c ne le sont pas.

(b) Étant donné un arbre binaire, la tâche consiste à retourner l'arbre binaire dans la direction droite, c'est-à-dire dans le sens des aiguilles d'une montre : le nœud le plus à gauche devient la racine de l'arbre inversé et son parent devient son enfant droit et le frère droit devient son enfant gauche et la même chose doit être faite pour tous les nœuds les plus à gauche de manière récursive. Écrivez un code qui implémente **tour à droite** pour un arbre binaire arbitraire. Testez le code sur l'exemple de la figure 2, puis donnez le résultat du code appliqué à l'arbre de droite de la figure 2. Utilisez votre code de l'exercice (a) pour trouver tous les cousins dans les trois arbres.

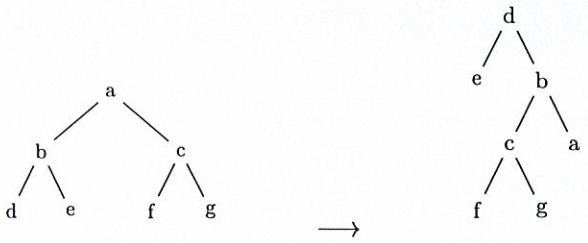

Figure 2: Exemple : Tour à droite

(c) On nous donne un arbre de taille n en tant que tableau $parent[0..n-1]$ où chaque index i dans le $parent[]$ représente un nœud et la valeur à i représente le parent immédiat de ce nœud. Pour le nœud racine, la valeur sera -1. Trouvez la hauteur de l'arbre générique en fonction des liens parents.

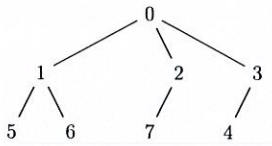

Figure 3: Arbre pour $parent[] = \{-1, 0, 0, 0, 3, 1, 1, 2\}$, avec hauteur 2.

Écrire un code qui trouve la hauteur d'un arbre K-aire arbitraire et l'utiliser pour trouver la hauteur de $parent[] = \{-1, 0, 1, 1, 1, 3, 3, 4, 7, 7, 9\}$