Les deux problèmes sont totalement indépendants. Les candidats doivent rédiger les questions dans l'ordre de l'énoncé ; des réponses précises sont attendues.

Les algorithmes demandés seront décrits en français ; les programmes seront rédigés en langage Caml ou en langage Pascal. Les candidats indiqueront au tout début de leur copie le langage de programmation qu'ils ont choisi d'utiliser. Une approche modulaire est vivement conseillée, comme l'a conseillé René DESCARTES : «Diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre».

Premier problème

I Notations

■ $|E|$ désigne le cardinal de l'ensemble fini E.

■ $\lfloor x \rfloor$ désigne la partie entière inférieure du réel x, et $\lceil x \rceil$ sa partie entière supérieure : ce sont les relatifs définis respectivement par $\lfloor x \rfloor \leqslant x < \lfloor x \rfloor + 1$ et $\lceil x \rceil - 1 < x \leqslant \lceil x \rceil$.

■ $[\![p, q]\!]$ désigne l'intervalle discret délimité par les naturels p et q : $[\![p, q]\!] = \{k \in \mathbb{N} \mid p \leqslant k \leqslant q\}$.

II Le problème du monnayeur

■ On s'intéresse au problème suivant, que l'on rencontre lors de la conception d'un monnayeur : comment rendre la monnaie, en utilisant le plus petit nombre possible de pièces ?

■ Formalisons ce problème : soit $(c_i)_{1 \leqslant i \leqslant m}$ une liste d'entiers non nuls, vérifiant

$$c_1 > c_2 > \ldots > c_{m-1} > c_m = 1$$

Nous utiliserons le terme *système* pour désigner une telle liste. Les c_i sont les *valeurs faciales* des pièces en service. Par exemple, si l'on omet les pièces de valeur inférieure à 1 franc, le système utilisé en France est $(20, 10, 5, 2, 1)$.

■ Pour chaque $i \in [\![1, m]\!]$, on dispose d'une quantité arbitrairement grande de pièces de valeur c_i. Soit x un entier (le montant à rendre). Une *représentation* de x dans le système c est un m-uplet $(k_i)_{1 \leqslant i \leqslant m}$ de naturels vérifiant $x = \sum_{1 \leqslant i \leqslant m} k_i c_i$: k_i est donc le nombre de pièces de valeur c_i que l'on doit rendre. Pour épargner les poches des clients, on souhaite minimiser le *poids* de cette représentation, c'est-à-dire le nombre $\|k\| = \sum_{1 \leqslant i \leqslant m} k_i$ de pièces rendues.

■ On peut considérer les différents m-uplets utilisés comme des vecteurs de $\mathbb{R}^m$, muni de sa structure euclidienne canonique. Si k est une représentation de x dans le système c, alors $x = k \cdot c$ et $\|k\| = k \cdot \mathbf{1}$, où $\mathbf{1}$ désigne le m-uplet dont toutes les composantes sont égales à 1.

——— Note à l'attention des candidats qui ont choisi le langage Caml. ———

■ Nous utiliserons des listes d'entiers pour représenter aussi bien un système (liste de valeurs de pièces), qu'une représentation d'un montant dans ce système.

——— Fin de la note à l'attention des candidats qui ont choisi le langage Caml. ———

——— La question suivante est destinée aux candidats qui ont choisi le langage Caml. ———

Question 1 • Rédigez en Caml une fonction de signature

```
est_un_systeme : int list -> bool
```

spécifiée comme suit : `est_un_systeme c` indique si la liste c est bien un système. Par exemple, `est_un_systeme [5;2;1]` rendra la valeur `true`, tandis que `est_un_systeme [5;7;1]` rendra la valeur `false`, tout comme le fera `est_un_systeme [7;5;2]`.

Note à l'attention des candidats qui ont choisi le langage Pascal.

■ Nous utiliserons des listes d'entiers pour représenter aussi bien un système (liste de valeurs de pièces), qu'une représentation d'un montant dans ce système. Le type utilisé est défini comme suit :

```
type Liste = ^Cellule;
type Cellule = record
 contenu : integer;
 suivant : Liste;
end;
```

La liste vide est représentée par la constante `nil`. Nous disposons d'une procédure et d'une fonction dont les en-têtes sont :

```
procedure pCons(x:integer; var c:Liste);
function fCons(x:integer; c:Liste):Liste;
```

spécifiées comme suit : `Cons(x,c)` ajoute en tête de la liste `c` une cellule dont le champ `contenu` contient `x` ; `fCons(x,c)` construit une liste dont la première cellule contient, dans son champ `contenu` la valeur de `x` et dans son champ `suivant` le pointeur `c`. Le programme suivant construit les listes $\ell_1 = (7,4,1)$ et $\ell_2 = (2,0,3)$, qui serviront dans les questions de programmation :

```
var l1,l2: Liste;

begin
 l1 := nil; pCons(1,l1); pCons(4,l1); pCons(7,l1);
 l2 := fCons(2,fCons(0,fCons(3,nil)));
end.
```

Fin de la note à l'attention des candidats qui ont choisi le langage Pascal.

La question suivante est destinée aux candidats qui ont choisi le langage Pascal.

Question 2 • Rédigez en Pascal une fonction d'en-tête

```
function est_un_systeme(c:Liste):boolean;
```

spécifiée comme suit : `est_un_systeme(c)` indique si la liste c est bien un système. Par exemple, `est_un_systeme(l1)` rendra la valeur `true`, tandis que `est_un_systeme(l2)` rendra la valeur `false`.

III Représentations de poids minimal

■ Soient $c = (c_i)_{1 \leqslant i \leqslant m}$ un système et $x \geqslant 0$. Nous notons $M_c(x)$ (ou même $M(x)$ lorsqu'aucune ambiguïté n'est à craindre) le plus petit nombre de pièces nécessaires pour représenter x dans le système c :

$$M(x) = \min\{\|k\| \mid k \in \mathbb{N}^m \text{ et } k \cdot c = x\}$$

Nous nous intéresserons aux représentations de poids minimal de x : ce sont les représentations k telles que $\|k\| = M(x)$. Pour alléger la rédaction, nous parlerons simplement de *représentations minimales*.

Question 3 • Prouvez l'encadrement $\lceil x/c_1 \rceil \leqslant M(x) \leqslant x$.

Question 4 • Exhibez un système c et un nombre $x > 0$ possédant plusieurs représentations minimales dans le système c.

Question 5 • Soient c un système et $x > 0$. Montrez que $M(x) \leqslant 1 + M(x - c_j)$ pour tout indice j tel que $c_j \leqslant x$.

Question 6 • Soient c un système et $x > 0$. Montrez que $M(x) = 1 + M(x - c_j)$ si et seulement si il existe une représentation minimale de x, faisant intervenir c_j.

Question 7 • Soit $x > 1$; on note s le plus petit indice i tel que $c_i \leqslant x$. On a donc $c_s \leqslant x$; et, soit $s = 1$, soit $s > 1$ et $c_{s-1} > x$. Justifiez l'égalité

$$M(x) = 1 + \min_{s \leqslant i \leqslant m} M(x - c_i)$$

Question 8 • Montrez que l'on peut construire la liste des valeurs de $M(y)$ pour $y \leqslant x$, pour un coût $\mathcal{O}(mx)$.

——— La question suivante est destinée aux candidats qui ont choisi le langage Caml. ———

Question 9 • Rédigez en Caml une fonction de signature

```
poids_minimaux : int -> int list -> int list
```

spécifiée comme suit : `poids_minimaux x c` construit la liste des valeurs de $M_c(y)$, pour $1 \leqslant y \leqslant x$. Par exemple, `poids_minimaux 5 [5;2;1]` rendra la liste `[1;1;2;2;1]`. Vous utiliserez la formule de la question 7.

——— La question suivante est destinée aux candidats qui ont choisi le langage Pascal. ———

Question 10 • Rédigez en Pascal une fonction d'en-tête

```
function poids_minimaux(x:integer; c:Liste):Liste;
```

spécifiée comme suit : `poids_minimaux(x,c)` rend une liste donnant les valeurs de $M_c(y)$, pour $1 \leqslant y \leqslant x$. Par exemple, `poids_minimaux(6,11)` rendra la liste $(1, 2, 3, 1, 2, 3)$. Vous utiliserez la formule de la question 7. Vous pourrez employer librement la constante `mMax`, le type `vecteur` et la fonction `vecteur_vers_Liste`, qui pourraient être définis ainsi :

```
const mMax = 1000; (* longueur maximale d'un vecteur *)

type vecteur = array[1..mMax] of integer;

function vecteur_vers_Liste(v:vecteur; r:integer):Liste;
var p:Liste;

begin
 p := nil;
 for k = r downto 1 do pCons(v[k],p);
 vecteur_vers_Liste := p;
end; (* vecteur_vers_Liste *)
```

Ainsi, `vecteur_vers_Liste(v,r)` construit la liste des $(v_i)_{1 \leqslant i \leqslant r}$ sous réserve que l'encadrement $1 \leqslant r \leqslant$ `mMax` soit vérifié. Si $r \leqslant 0$, elle rend la liste vide ; si $r >$ `mMax`, le résultat n'est pas défini.

IV L'algorithme glouton

G ■ L'algorithme *glouton* consiste à choisir le plus grand $c \leqslant x$, puis à rendre le montant $x - c$. Par exemple, avec le système $(10, 5, 2, 1)$ l'algorithme glouton décomposera le montant 27 en $27 = 10 + 10 + 5 + 2$. Avec le formalisme proposé, $m = 4$, $c_1 = 10$, $c_2 = 5$, $c_3 = 2$, $c_4 = 1$; la solution fournie par l'algorithme glouton est le quadruplet $(k_1 = 2, k_2 = 1, k_3 = 1, k_4 = 0)$. Le fonctionnement de l'algorithme glouton peut être accéléré avec la remarque suivante : notant $q = \lfloor x/c_1 \rfloor$, cet algorithme rend q pièces de valeur c_1, puis rend le montant $x - qc_1$ en utilisant le système $(c_i)_{2 \leqslant i \leqslant m}$.

——— La question suivante est destinée aux candidats qui ont choisi le langage Caml. ———

Question 11 • Rédigez en Caml une fonction de signature

```
glouton : int -> int list -> int list
```

spécifiée comme suit : `glouton x c` construit la représentation de x dans le système c, en appliquant l'algorithme glouton. Par exemple, `glouton 13 [5;2;1]` rendra la liste `[2;1;1]`.

——— La question suivante est destinée aux candidats qui ont choisi le langage Pascal. ———

Question 12 • Rédigez en Pascal une fonction d'en-tête

```
function glouton(x:integer; c:Liste):Liste;
```

spécifiée comme suit : `glouton(x,c)` rend une liste donnant la représentation de x dans le système c, en appliquant l'algorithme glouton. Par exemple, `glouton(17,11)` rendra une liste identique à `12`.

■ Soient $c = (c_i)_{1 \leqslant i \leqslant m}$ un système et $x \geqslant 0$. Nous noterons $\Gamma_c(x)$ la représentation gloutonne de x dans le système c, et $G_c(x)$ (ou même $G(x)$ lorsqu'aucune ambiguïté n'est à craindre) le nombre de pièces utilisées par l'algorithme glouton : $G_c(x) = \|\Gamma_c(x)\|$.

■ Nous dirons qu'un système $c = (c_i)_{1 \leqslant i \leqslant m}$ est *canonique* si l'algorithme glouton donne toujours une représentation minimale ; on a donc $M(x) = G(x)$ quel que soit x.

Question 13 • Montrez que tout système $c = (c_1, c_2)$ de deux valeurs est canonique.

Question 14 • Exhibez un système $c = (c_1, c_2, c_3)$ non canonique.

Question 15 Montrez que tout système $c = (c_i)_{1 \leqslant i \leqslant m}$ vérifiant $c_i \geqslant 2c_{i+1}$ pour tout $i \in [\![1, m-1]\!]$ est canonique.

■ Le résultat précédent montre que le système utilisé en France est canonique : si nous prenons comme unité la pièce de 5 centimes, ce système est $(400, 200, 100, 40, 20, 10, 4, 2, 1)$.

Question 16 • Avant la réforme de 1971 (introduisant un système décimal), le Royaume-Uni utilisait le système $(30, 24, 12, 6, 3, 1)$. Montrez que ce système n'est pas canonique.

Question 17 • Considèrons un monnayeur basé sur un système canonique. Donnez un exemple de situation où il est souhaitable que ce monnayeur n'applique pas l'algorithme glouton.

V L'algorithme de Kozen et Zaks

■ Nous nous proposons de décrire un algorithme efficace permettant de déterminer si un système c est canonique. Nous dirons qu'un entier x est un *contre-exemple* pour c si $M(x) < G(x)$. Un système canonique n'admet donc aucun contre-exemple.

Question 18 • Soit c un système non canonique. Montrez qu'il existe un contre-exemple x vérifiant l'encadrement $c_{m-2} + 1 < x < c_1 + c_2$.

■ Nous allons montrer que l'intervalle de recherche des contre-exemples ne peut pas être diminué.

Question 19 • Soit $q \geqslant 2$. Montrez que le système $(q+1, q, 1)$ n'est pas canonique. Quel est le plus petit contre-exemple pour ce système?

Question 20 • Soit $q \geqslant 3$. Déterminez $\alpha(q) > q$ tel que le système $(\alpha(q), q, 1)$ ne soit pas canonique, et admette $\alpha(q)+2$ comme plus petit contre-exemple.

■ Le résultat de la question 18 nous donne un algorithme déterminant si un système est canonique : il suffit de les rechercher dans l'intervalle discret $[\![c_{m-2}+2, c_1+c_2-1]\!]$. Ceci nécessite la construction des représentations minimales de chacun des éléments de cet intervalle.

■ Nous allons étudier un algorithme plus efficace, dû à KOZEN et ZAKS ; leur méthode repose sur la notion de *témoin*. Nous dirons qu'un entier x est un témoin pour le système $c = (c_i)_{1 \leqslant i \leqslant m}$ s'il existe un indice i tel que $G(x - c_i) < G(x) - 1$.

Question 21 • Montrez que tout témoin est un contre-exemple.

Question 22 • Montrez que le résultat de la question 21 n'admet pas de réciproque ; vous exhiberez un système $c = (s, t, 1)$ et un entier x qui est un contre-exemple, mais pas un témoin pour c.

Question 23 • Montrez que si le système c admet des contre-exemples, alors le plus petit d'entre eux est un témoin.

■ Il résulte de cette étude que, pour savoir si un système est canonique, il suffit de vérifier l'inégalité $G(x) \leqslant G(x - c_i) + 1$ pour tout $x \in [\![c_{m-2}+2, c_1+c_2-1]\!]$, et tout indice $i \in [\![1, m]\!]$ tel que $c_i < x$. S'est le principe de l'algorithme de KOZEN et ZAKS.

——— La question suivante est destinée aux candidats qui ont choisi le langage Caml. ———

Question 24 • Rédigez en Caml une fonction de signature

```
kozen_zaks : int list -> bool
```

spécifiée comme suit : `kozen_zaks c` indique si la liste c est canonique.
Par exemple, `kozen_zaks [5;2;1]` rendra la valeur `true`, tandis que `kozen_zaks [6;5;1]` rendra la valeur `false`. Vous utiliserez l'algorithme de KOZEN et ZAKS.

——— La question suivante est destinée aux candidats qui ont choisi le langage Pascal. ———

Question 25 • Rédigez en Pascal une fonction d'en-tête

```
function kozen_zaks(c:Liste):boolean;
```

spécifiée comme suit : `kozen_zaks(c)` indique si la liste c est canonique.
Par exemple, `kozen_zaks(l1)` rendra la valeur `true`. Vous utiliserez l'algorithme de KOZEN et ZAKS.

Question 26 • Montrez que le coût de l'algorithme de KOZEN et ZAKS peut être exponentiel, par rapport au nombre m de pièces du système.

Question 27 • Montrez qu'un système $(s, t, 1)$ est non canonique si et seulement si, notant q le quotient et r le reste dans la division euclidienne de s par t, on a $0 < r < t - q$.

VI L'algorithme de Pearson

■ Dans cette partie, nous allons étudier un algorithme, dû à PEARSON, dont le coût est polynomial par rapport au nombre m de pièces du système.

■ Soient $u = (u_i)_{1 \leqslant i \leqslant m}$ et $v = (v_i)_{1 \leqslant i \leqslant m}$ deux m-uplets de réels. Nous noterons $u \leqslant v$ si $u_i \leqslant v_i$ pour tout indice $i \in [\![1, m]\!]$; $\mathbb{R}^m$ est ainsi muni d'un ordre partiel. Nous noterons $u \prec v$ s'il existe un indice $j \in [\![1, m]\!]$ tel que $u_i = v_i$ pour $1 \leqslant i < j$, et $u_j < v_j$. Nous noterons $u \preceq v$ si $u = v$ ou $u \prec v$: nous avons ainsi défini sur $\mathbb{R}^m$ l'ordre *lexicographique* ; cet ordre est total.

Question 28 • Soient $c = (c_i)_{1 \leqslant i \leqslant m}$ un système, et $x > 0$. Montrez que $\Gamma_c(x)$ est, pour l'ordre lexicographique, le plus grand élément de l'ensemble des représentations de x.

Question 29 • Soient x et y deux réels. Montrez que $x < y$ implique $\Gamma_c(x) \prec \Gamma_c(y)$.

Question 30 • Soient u et v deux m-uplets de naturels. On suppose $u \leqslant v$ et $v = \Gamma_c(v \cdot \mathbf{1})$. Montrez que $u = \Gamma_c(u \cdot \mathbf{1})$.

■ Soient $c = (c_i)_{1 \leqslant i \leqslant m}$ un système, et $x > 0$. Parmi les représentations de x de poids minimal dans le système c, nous noterons $\Omega_c(x)$ celle qui est la plus grande pour l'ordre lexicographique ; nous dirons que $\Omega_c(x)$ est *la représentation optimale* de x.

Question 31 • Soient u et v deux m-uplets de naturels. On suppose $u \leqslant v$ et $v = \Omega_c(v \cdot \mathbf{1})$. Montrez que $u = \Omega_c(u \cdot \mathbf{1})$.

■ Dans toute la suite, c désigne un système non canonique, et ξ le plus petit contre-exemple pour ce système. Notons $g = \Gamma_c(\xi)$ et $\omega = \Omega_c(\xi)$.

Question 32 • Montrez que $g_i \omega_i = 0$ pour tout $i \in [\![1, m]\!]$.

■ On note i le plus petit indice tel que $\omega_i \neq 0$, et j le plus grand indice tel que $\omega_j \neq 0$.

Question 33 • Prouvez que i est au moins égal à 2.

Question 34 • Soit $\gamma = \Gamma_c(c_{i-1} - 1)$. Montrez que $\omega_k = \gamma_k$ pour $1 \leqslant k < j$, et $\omega_j = \gamma_j + 1$.

Question 35 • Déduisez de ce dernier résultat un algorithme déterminant si un système c donné est canonique, pour un coût $\mathcal{O}(m^3)$.

——— La question suivante est destinée aux candidats qui ont choisi le langage Caml. ———

Question 36 • Rédigez en Caml une fonction de signature

```
pearson : int list -> bool
```

spécifiée comme suit : `pearson c` indique si la liste c est canonique.
Par exemple, `pearson [5;2;1]` rendra la valeur `true`, tandis que `pearson [6;5;1]` rendra la valeur `false`. Vous utiliserez l'algorithme de PEARSON.

——— La question suivante est destinée aux candidats qui ont choisi le langage Pascal. ———

Question 37 • Rédigez en Pascal une fonction d'en-tête

```
function pearson(c:Liste):boolean;
```

spécifiée comme suit : `pearson(c)` indique si la liste c est canonique.
Par exemple, `pearson(l1)` rendra la valeur `true`. Vous utiliserez l'algorithme de PEARSON.
