Exercices de programmation

Julien Reichert

2024/2025

Ce document est une liste d'exercices de programmation faisant office de « minimum syndical » en termes de maîtrise de la syntaxe des langages de programmation au programme de classes préparatoires (Python pour le tronc commun, OCaml pour l'option informatique et l'informatique de MP2I, C pour l'informatique de MP2I et SQL pour le tronc commun et l'informatique de MP2I). Il est écrit spécialement pour la classe de MP2I, mais peut être utilisé par d'autres classes en adaptant la chronologie à la progression prévue par le programme officiel, ainsi que le langage.

Les exercices qui y figurent sont à faire chez soi, en faire la correction en classe n'est pas systématique, bien qu'on puisse imaginer des séances spécialement dédiées à cette liste.

Il est possible que des exercices aient des prérequis mathématiques. Normalement, ils sont placés dans une période où l'avancée du cours dans les deux matières est suffisante.

Table des matières

1 Les exercices à maîtriser avant la rentrée [Python]

Toutes les fonctions de cette section sont à écrire en Python.

Exercice 0 : Écrire une fonction `somme` prenant en argument deux nombres et retournant leur somme. Vérifier que `somme(somme(10, 11), somme(11, 10))` fonctionne.

Exercice 1 : Écrire une fonction `milieu` prenant en argument une liste et retournant l'élément au milieu de la liste. S'il y en a deux, prendre celui de gauche.

Exercice 2 : Écrire une fonction `som` prenant en argument une liste et retournant la somme des éléments aux indices pairs et uniquement de ceux-ci.

Exercice 3 : Écrire une fonction `gau` prenant en argument une liste et retournant la sous-liste constituée de sa première moitié. Si la taille est impaire, on incorpore l'élément du milieu.

Exercice 4 : Écrire une fonction `majobool` prenant en argument une liste de 0 et de 1 et retournant l'élément le plus fréquent.

Exercice 5 : Écrire une fonction `maxi` prenant en argument une liste de nombres et retournant son maximum.

Exercice 6 : Écrire une fonction `viceminor` prenant en argument une liste de nombres et retournant son deuxième plus petit élément.

Exercice 7 : Écrire une fonction `lesplus` prenant en argument une liste d'entiers et retournant la liste des entiers positifs qui y figurent, dans le même ordre.

Exercice 8 : Écrire une fonction `trouvezero` prenant en argument une liste d'entiers et retournant la liste des indices où l'entier zéro y figure, dans l'ordre croissant.

Exercice 9 : Écrire une fonction `riorim` prenant en argument une liste d'entiers et retournant la liste dans l'ordre inverse.

2 Les exercices de la première période [C]

Toutes les fonctions de cette section sont à écrire en C.

Exercice 10 : Écrire une fonction `fraction` prenant en argument deux entiers dont le deuxième est non nul et retournant le quotient du premier par le deuxième en tant que flottant.

Exercice 11 : Écrire une fonction `bissextile` prenant en argument un entier supérieur à `2000` et retournant un booléen déterminant si cet entier correspond à une année bissextile.

Exercice 12 : Écrire une fonction `evalue` prenant en argument un tableau de flottants, sa taille et un autre flottant et retournant le résultat de l'évaluation de la fonction polynomiale associé au tableau sur le flottant en troisième argument. Le premier indice correspond au plus bas degré.

Exercice 13 : Écrire une fonction `ecartmin` prenant en argument un tableau d'entiers de taille au moins deux (pas à vérifier) et sa taille et retournant le plus petit écart en valeur absolue entre deux valeurs consécutives du tableau.

Exercice 14 : Écrire une fonction `extraire` prenant en argument une chaîne de caractères et retournant le premier entier rencontré dans la chaîne, positif ou négatif.

Exercice 15 : Écrire une fonction `sommemajo` prenant en argument un tableau de flottants et sa taille et retournant la somme des éléments positifs du tableau ou la somme de ses éléments négatifs en fonction du signe apparaissant le plus souvent. En cas d'égalité d'effectif on retournera zéro, et les éléments nuls sont ignorés.

Exercice 16 : Écrire une fonction `nbdiff` prenant en argument une chaîne de caractères et retournant le nombre de caractères différents qui la composent.

Exercice 17 : Écrire une fonction `palindromeint` prenant en argument un entier positif et déterminant s'il s'agit d'un palindrome en base dix.

Exercice 18 : Écrire une fonction `palindrome2` prenant en argument une chaîne de caractères et déterminant si, en ne considérant que les lettres, la chaîne forme un palindrome, en considérant une minuscule comme égale à la majuscule correspondante.

Exercice 19 : Écrire une fonction `sommeentiers` prenant en argument une chaîne de caractères et retournant la somme des entiers rencontrés dans la chaîne, positifs ou négatifs.

3 Les exercices des vacances de la Toussaint [C]

Toutes les fonctions de cette section sont à écrire en C.

Exercice 20 : Écrire une fonction `spitze` prenant en argument un tableau de booléens et sa taille et retournant le nombre de booléens égaux au premier élément du tableau jusqu'au premier qui soit différent. La valeur minimale possible sera un.

Exercice 21 : Écrire une fonction `nul` prenant en argument un tableau de booléens et sa taille et déterminant si jusqu'à la dernière apparition d'un `true` il y a du début à chaque indice constamment au moins autant de `true` que de `false`.

Exercice 22 : Écrire une fonction `modfloat` prenant en argument deux flottants et retournant le reste dans la pseudo-division euclidienne du premier par le deuxième.

Exercice 23 : Écrire une fonction `minmax` prenant en argument un tableau d'entiers et sa taille et retournant le minimum et le maximum du tableau.

Exercice 24 : Écrire une fonction `medianemax` prenant en argument un tableau d'entiers et sa taille et retournant l'indice médian parmi les occurrences du maximum du tableau. S'il y en a un nombre pair, on retournera l'indice médian « de gauche ».

Exercice 25 : Écrire une fonction `derive` prenant en argument un tableau de flottants et sa taille, ce tableau représentant un polynôme avec le coefficient constant en première position, et mutant ce tableau de sorte qu'il représente la dérivée du polynôme initial. On ne se souciera pas de la différence de taille que cela implique.

Exercice 26 : Écrire une fonction `multpol` prenant en argument deux tableaux de flottants et leurs tailles, ces tableaux représentant des polynômes avec les coefficients constants en première position, et retournant le produit de ces polynômes.

Exercice 27 : Écrire une fonction `cesar` prenant en argument une chaîne de caractères et deux entiers et retournant la chaîne correspondant au cryptage de César par la fonction affine (déclencher une erreur si elle n'est pas bijective) dont les coefficients sont les entiers correspondants. Les capitales resteront des capitales, les minuscules aussi, et les autres caractères ne subiront pas de transformation.

Exercice 28 : Écrire une fonction `auguste` prenant en argument un entier, dont on vérifiera qu'il est entre `1` et `3999`, et retournant la chaîne de caractères correspondant à cet entier écrit en chiffres romains.

Exercice 29 : Écrire une fonction `ecart` prenant en argument un tableau d'entiers et sa taille (au moins deux) et retournant deux éléments arbitraires parmi les moins fréquents du tableau : soit ils n'apparaissent qu'une fois, soit l'un n'apparaît qu'une fois et l'autre minimise le nombre d'occurrences sinon, soit on donne deux occurrences du même élément parmi ceux qui minimisent le nombre d'occurrences.

4 Les exercices de la deuxième période [Ocaml]

Toutes les fonctions de cette section sont à écrire en OCaml.

Exercice 30 : Écrire une fonction `plusetfois` prenant en argument deux entiers et retournant un couple constitué de leur somme et leur produit.

Exercice 31 : Écrire une fonction `ppp2` prenant en argument un entier que l'on supposera positif et retournant la plus petite puissance de deux supérieure ou égale à ce nombre, en notant bien que ce n'est pas la puissance à laquelle élever deux mais le résultat de l'opération que l'on veut.

Exercice 32 : Écrire une fonction `leeloo` prenant en argument une liste et retournant son cinquième élément s'il existe (sinon comportement au choix).

Exercice 33 : Écrire une fonction `coupe` prenant en argument une liste et retournant un couple de listes contenant dans le même ordre les éléments de la liste fournie répartis alternativement en commençant par la première liste du couple.

Exercice 34 : Écrire une fonction `colle` prenant en argument deux listes dont on supposera qu'elles sont de même taille et retournant la liste obtenue en prenant alternativement un élément de chaque liste.

Exercice 35 : Écrire une fonction `compte` prenant en argument une liste et une valeur du type des éléments de la liste et retournant le nombre d'occurrences de la valeur dans la liste.

Exercice 36 : Écrire une fonction `sommesi` prenant en argument une liste d'entiers et un prédicat sur les entiers (c'est-à-dire une fonction prenant en entier et retournant un booléen) et retournant la somme des éléments de la liste vérifiant le prédicat (donc celui-ci renvoie vrai) et seulement eux.

Exercice 37 : Écrire une fonction `majopred` prenant en argument une liste et un prédicat sur les éléments de la liste et retournant un booléen déterminant si au moins la moitié des éléments de la liste vérifient le prédicat.

Exercice 38 : Écrire une fonction `lexico` prenant en argument deux listes et retournant un booléen indiquant si la première est inférieure à la deuxième dans un ordre lexicographique (le premier élément strictement inférieur à l'élément à la même position entre les listes détermine le résultat final, et si une liste est un préfixe de l'autre elle y est inférieure).

Exercice 39 : Écrire une fonction `aplatir` prenant en argument une liste de listes et retournant la liste contenant tous les éléments de ces listes à la suite.

5 Les exercices des vacances de Noël [Ocaml]

Toutes les fonctions de cette section sont à écrire en OCaml.

Exercice 40 : Écrire une fonction `zip` prenant en argument deux tableaux de même taille et retournant le tableau des couples formés par les éléments apparaissant à chaque indice respectif des tableaux, dans le même ordre.

Exercice 41 : Écrire une fonction `tirette` prenant en argument un tableau d'entiers de taille paire et retournant le tableau des sommes de deux éléments appariés depuis l'extérieur vers l'intérieur (premier + dernier, deuxième + avant-dernier, etc.).

Exercice 42 : Écrire une fonction `quarante_deux` prenant en argument un tableau d'entiers et déterminant s'il y a exactement quarante-deux indices multiples de quarante-deux où figurent des quarante-deux dans le tableau.

Exercice 43 : Écrire une fonction `croissant` prenant en argument un tableau et déterminant s'il est croissant.

Exercice 44 : Écrire une fonction `bitonique` prenant en argument un tableau et déteminant s'il est croissant puis décroissant (la phase croissante ou la phase décroissante peut être vide, tant qu'après la décroissance il n'y ait pas de croissance).

Exercice 45 : Écrire une fonction `psrn` prenant en argument deux tableaux de flottants et retournant la somme indice par indice du produit des éléments de même indice dans les deux tableaux.

Exercice 46 : Écrire une fonction `rotation` prenant en argument un tableau de tableaux tous de même taille et retournant ce tableau « pivoté d'un quart de tour dans le sens horaire ».

Exercice 47 : Écrire une fonction `plsci` prenant en argument un tableau et retournant un couple donnant le début et la taille de la plus longue séquence d'éléments consécutifs identiques dans le tableau. En cas d'égalité de tailles, on prendra la première.

Exercice 48 : Écrire une fonction `sorteren` prenant en argument un tableau dont les éléments sont des `0`, `1` ou `2` et mutant le tableau afin de le trier dans l'ordre croissant.

Exercice 49 : Écrire une fonction `dominant` prenant en argument une chaîne de caractères dont on admettra qu'ils proviennent de la table ASCII non étendue et retournant le caractère le plus fréquent. En cas d'égalité n'importe lequel pourra être retourné.

6 Les exercices de la troisième période [C et OCaml]

Toutes les fonctions de cette section sont à écrire en C et en OCaml. Si un tableau doit être un argument, sa taille est à mettre en argument aussi en C uniquement.

Exercice 50 : Écrire une fonction `tables` prenant en argument un entier `n` supérieur ou égal à six et retournant le couple `(a, b)` tel que `n` soit égal à quatre fois `a` plus trois fois `b`, avec `b` valant au plus trois.

Exercice 51 : Écrire une fonction `q1` prenant en argument un tableau (en C : forcément de flottants) de taille un multiple de quatre plus un et retournant son premier quartile.

Exercice 52 : Écrire une fonction `minecart` prenant en argument un tableau d'entiers et déterminant le plus petit écart entre un élément du tableau et l'indice où il se situe.

Exercice 53 : Écrire une fonction `premiercommepremier` prenant en argument un tableau (en C : forcément d'entiers) et retournant l'indice du premier élément au-delà du premier égal au premier. S'il n'y en a pas, on retournera `-1`.

Exercice 54 : Écrire une fonction `premiercommeavant` prenant en argument un tableau (en C : forcément d'entiers) et retournant l'indice du premier élément égal à un élément à sa gauche et l'indice de celui-ci, **dans cet ordre**. S'il n'y en a pas, on retournera `(-1, -1)`.

Exercice 55 : Écrire une fonction `equilibre` prenant en argument un tableau d'entiers et déterminant si tous les éléments ont le même nombre d'occurrences.

Exercice 56 : Écrire une fonction `premierabsent` prenant en argument un tableau d'entiers et déterminant le plus petit entier naturel absent du tableau.

Exercice 57 : Écrire une fonction `rpz` prenant en argument une chaîne de caractères et déterminant le nombre de fois qu'on peut écrire le nom du plus beau département de France (dont le numéro est celui de cette question) à partir des lettres de cette chaîne, quelle que soit la casse.

Exercice 58 : Écrire une fonction `decomp` prenant en argument un entier naturel non nul et retournant le tableau de ses facteurs premiers, répétés autant de fois que la valuation associée (en OCaml : la liste des couples formés par les diviseurs premiers et la valuation associée).

Exercice 59 : Écrire une fonction `evaluationHL` prenant en argument une main de treize cartes, la structure de carte étant définie avec deux champs de type chaîne de caractères (valeur et couleur), et déterminant le nombre de points HL de cette main au bridge (préciser en commentaire à quel point l'attribution des points L est stricte...).

7 Les exercices des vacances d'hiver [C et OCaml]

Toutes les fonctions de cette section sont à écrire en C et en OCaml. Si un tableau doit être un argument, sa taille est à mettre en argument aussi en C uniquement.

Exercice 60 : Écrire une fonction `mcnuggets` prenant en argument deux entiers naturels et déterminant le plus grand entier (erreur si infini) qui ne peut pas s'écrire comme une combinaison linéaire à coefficients dans $\mathbb{N}$ de ces deux entiers.

Exercice 61 : Écrire une fonction `existesomme` prenant en argument un tableau d'entiers et déterminant si un des éléments du tableau est égal à la somme de deux autres éléments. Attention : les trois indices doivent être différents deux à deux.

Exercice 62 : Écrire une fonction `fincommune` prenant en argument deux chaînes de caractères contenant uniquement des lettres minuscules (pas à vérifier) et retournant le nombre de lettres communes à ces mots de manière consécutive à partir de la fin.

Exercice 63 : Écrire une fonction `simplifiable` prenant en argument un tableau de fractions (structures avec deux champs entiers, numérateur et dénominateur) et déterminant s'il existe deux fractions égales. Erreur si un des dénominateurs est nul.

Exercice 64 : Écrire une fonction `miroirlc` prenant en argument une liste chaînée modifiable (en C : forcément d'entiers) et retournant (c'est le cas de le dire) le miroir de la liste chaînée, en tant que liste chaînée aussi. On implémentera tout d'abord la structure et on ne se servira que des opérations élémentaires de l'interface. L'argument sera laissé intact à la fin.

Exercice 65 : Écrire une fonction `sommeminmax` résolvant le problème suivant : on considère un tableau d'entiers de taille paire (pas à vérifier) ; étant donné une manière de rassembler les éléments deux par deux, on associe à cet appariement la somme des éléments maximaux de chaque couple, et on cherche la plus petite telle somme.

Exercice 66 : Écrire une fonction `medianefreq` prenant en argument un tableau (en C : forcément d'entiers) et retournant l'indice médian parmi les occurrences de l'élément le plus fréquent du tableau, « de gauche » s'il y en a un nombre pair. À égalité de nombre d'occurrences, on gardera l'élément dont la somme des indices est minimale, et en cas de nouvelle égalité on choisira la position médiane la plus petite.

Exercice 67 : Écrire une fonction `niederschaeffolsheim` prenant en argument une chaîne de caractères contenant uniquement des lettres et des espaces, sans jamais deux espaces consécutives (rien de tout cela n'est à vérifier) et retournant la taille maximale d'un mot (sous-chaîne délimitée par des espaces ou la fin du mot).

Exercice 68 : Écrire une fonction `kneckes` prenant en argument une chaîne de caractères avec les mêmes propriétés que précédemment et retournant la chaîne obtenue en glissant avant chaque mot (définition identique) « d'indice un nombre premier » le mot `"hopla"`.

Exercice 69 : Écrire une fonction `crib` prenant en argument une carte et une main en tant que tableau de quatre cartes (même type que pour l'exercice 59) et retournant le nombre de points obtenus par le donneur au jeu de cribbage avec cette main et cette carte annexe (dont les deux points si c'est un valet). On consultera les règles sur internet, et on n'hésitera pas à imprimer le détail des points en vue de déboguer la fonction.

8 Les exercices de la quatrième période [C et OCaml]

Toutes les fonctions de cette section sont à écrire en C et en OCaml. Si un tableau doit être un argument, sa taille est à mettre en argument aussi en C uniquement.

Exercice 70 : Écrire une fonction `cycle` prenant en argument un tableau d'entiers entre zéro et la taille du tableau moins un et déterminant s'il représente une permutation se limitant à un cycle.

Exercice 71 : Écrire une fonction `idem` prenant en argument un tableau d'entiers entre zéro et la taille du tableau moins un et déterminant s'il représente une fonction dont l'application répétée autant de fois que le nombre d'éléments du tableau correspond à l'identité.

Exercice 72 : Écrire une fonction `elaguer` prenant en argument une chaîne de caractères et retournant la même chaîne sans les espaces, tabulations ou sauts de ligne avant le premier autre caractère ou après le dernier autre caractère. Les enchaînements de tels caractères à l'intérieur seront aussi remplacés par une seule espace.

Exercice 73 : Écrire une fonction `nombrediviseurs` prenant en argument un entier naturel non nul et retournant son nombre de diviseurs positifs, premiers ou non.

Exercice 74 : Écrire une fonction `maxintersect` prenant en argument deux tableaux d'entiers représentant le même ensemble d'intervalles fermés, triés l'un par début croissant et l'autre par fin croissante (en C : on pourra considérer qu'il s'agit de versions aplaties en alternant les indices ; en OCaml : ce seront des tableaux de couples d'entiers), et retournant le nombre maximal d'intersections d'intervalles ayant lieu.

Exercice 75 : Écrire une fonction `ptm` prenant en argument un entier naturel et retournant une chaîne de caractères correspondant au terme de la suite de Prouhet-Thue-Morse dont l'indice est cet entier.

Exercice 76 : Écrire une fonction `zerosfact` prenant en argument un entier naturel et retournant le nombre de zéros à la fin de l'écriture en base dix de la factorielle de cet entier.

Exercice 77 : Écrire une fonction `hauteurarbre` prenant en argument un arbre d'arité quelconque (structure à définir comme en cours) et retournant sa hauteur.

Exercice 78 : Écrire une fonction `tessiture` prenant en argument un tableau de notes et déterminant le nombre de demi-tons entre la note la plus grave et la note la plus aigüe. Les notes sont des chaînes contenant le nom en minuscule et l'octave (sur un chiffre), par exemple `sol3`, `mib1` et `fad2`. Il peut y avoir deux noms pour la même note dans le tableau, mais ce ne seront que des notes existant réellement (inutile de le vérifier).

Exercice 79 : Écrire une fonction `diveucl` prenant en argument deux tableaux de flottants représentant des polynômes, le premier indice correspondant au coefficient constant, et retournant sous la même forme le quotient dans la division euclidienne du premier polynôme par le second.

9 Les exercices des vacances de Pâques [C et OCaml]

Toutes les fonctions de cette section sont à écrire en C et en OCaml. Si un tableau doit être un argument, sa taille est à mettre en argument aussi en C uniquement. Une « séquence » est un tableau en C ; en OCaml, c'est une liste ou un tableau, au choix.

Exercice 80 : Écrire une fonction `ppcm_ens` prenant en argument une séquence d'entiers naturels non nuls et retournant le PPCM global de cette séquence.

Exercice 81 : Écrire une fonction `subsetsum` prenant en argument une séquence d'entiers ainsi qu'un autre entier et déterminant si celui-ci peut s'écrire comme la somme d'une sous-séquence de la séquence.

Exercice 82 : Écrire une fonction `selectfrom` prenant en argument une séquence de tableaux (en C : d'entiers) tous de même taille ainsi qu'une séquence supplémentaire d'entiers entre zéro et la taille du tableau moins un (pas forcément monotone, mais on peut supposer qu'il n'y a pas de répétitions) et retournant une séquence formée des tableaux précédents dans le même ordre au sein de la séquence mais dont les éléments sont ceux aux indices précisés dans la séquence en respectant l'ordre d'énonciation.

Exercice 83 : Écrire une fonction `strahler` prenant en argument un arbre (type sans étiquette à définir, arité quelconque) et retournant son nombre de Strahler.

Exercice 84 : Écrire une fonction `dist_n` prenant en argument un graphe orienté (représentation au choix), un sommet du graphe et un entier naturel et retournant la séquence des sommets qui sont la destination d'au moins un chemin d'origine le sommet en argument et de longueur exactement l'entier en argument.

Exercice 85 : Écrire une fonction `unitaire` prenant en argument une séquence de flottants et retournant une séquence de flottants représentant un polynôme unitaire ayant exactement les éléments de la séquence en argument pour racine. Le premier élément de la séquence retournée correspondra au coefficient constant.

Exercice 86 : Écrire une fonction `avggroupby` prenant en argument une séquence de couples formés chacun par une chaîne de caractères et un nombre (entier ou flottant au choix, utiliser `struct` en C) et retournant une séquence de couples dont les premiers éléments sont toutes les chaînes de caractères rencontrées sans répétition et les deuxièmes éléments sont les moyennes (forcément en tant que flottants) des valeurs associées à la chaîne correspondante.

Exercice 87 : Écrire une fonction `facteurmax` prenant en argument une chaîne de caractères et localisant le plus long facteur de la chaîne apparaissant au moins deux fois. En cas d'égalité on prendra le premier à apparaître. La fonction devra retourner les deux premiers indices où le facteur commence.

Exercice 88 : Écrire une fonction `hyperplan` prenant en argument une séquence de $n-1$ vecteurs (séquences de flottants) en dimension n et retournant une équation de l'hyperplan qu'ils engendrent sous forme d'une séquence de taille n. Cette séquence devra avoir `1.0` pour premier élément non nul.

Exercice 89 : Écrire une fonction `belote` prenant en argument une chaîne de caractères, un entier et un tableau de huit tableaux de quatre cartes qui correspondent aux huit plis d'une donne de belote, chacun dans l'ordre des cartes jouées et déterminant si l'équipe qui a pris le contrat a gagné la donne, sachant que l'atout est la chaîne en argument et que le joueur qui a pris le contrat est, au premier pli, à l'indice en argument.

10 Les exercices de la cinquième période [SQL]

Pour tous les exercices, la consigne commence par « écrire une requête en SQL pour déterminer… ».

La base de données est la même pour tous les exercices, son contenu est détaillé ci-après et des explications figurent dans le devoir numéro 1 des PC en 2022/2023 :

- Table `Pieces`, dont les attributs sont `id_piece` (entier), `pays` (chaîne de caractères), `valeur` (flottant), `unite` (entier), `annee` (entier) et `valable` (booléen).
- Table `Unites`, dont les attributs sont `id_unite` (entier), `symbole` (chaîne de caractères) et `nom_unite` (chaîne de caractères).
- Table `Cours`, dont les attributs sont `monnaie` (entier) et `euros` (flottant).
- Table `Rangement`, dont les attributs sont `piece` (entier), `valise` (entier), `compartiment` (entier), `rangee` (entier) et `colonne` (entier).

Exercice 90 : Le nom de la monnaie dont le symbole est `"CHF"`.

Exercice 91 : Le nombre de pièces ayant actuellement cours légal dans ma collection.

Exercice 92 : Le pays d'où provient la pièce ayant cours légal avec la plus grande valeur faciale (en cas d'égalité, on acceptera une seule valeur mais on préfèrera l'ensemble des valeurs).

Exercice 93 : Le nom de l'unité monétaire dont j'ai le plus grand nombre de pièces au total (en comptant les doublons autant de fois que nécessaire).

Exercice 94 : Le nombre total de cases de rangement occupées par au moins une pièce.

Exercice 95 : Le plus grand nombre de pièces dans une même case de rangement.

Exercice 96 : Le nom de l'unité monétaire ayant cours légal pour laquelle l'ensemble des pièces dont je dispose a le plus de valeur en ramenant aux euros (comme des égalités sont improbables, on pourra ignorer celles-ci).

Exercice 97 : Le nombre d'unités monétaires dont le cours n'est pas précisé.

Exercice 98 : Le nombre total d'emplacements des compartiments en admettant que toutes les valises contiennent au moins une pièce et aient le même nombre de compartiments, que chaque compartiment ait le même nombre de rangées et qu'au moins une pièce figure dans la dernière rangée d'un compartiment, et que chaque rangée ait le même nombre de colonnes et qu'au moins une pièce figure dans la dernière colonne d'une rangée.

(En pratique ce n'est pas vrai, des compartiments ont des emplacements plus gros que d'autres, mais peu importe.)

Exercice 99 : Le nombre d'emplacements vides sous les mêmes conditions que pour l'exercice précédent.

11 Les exercices des vacances d'été [tous]

Exercice 100 : Écrire dans tous les langages enseignés un message différent exprimant le soulagement que l'année soit finie.