L'usage des calculatrices est interdit

Les élèves de mpsi3 ne font que les questions 1., 2., 3., 4., 5.

Problème

Le but de ce problème est de généraliser les jeux du type :

En intercalant uniquement des signes + et −, compléter : $9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1 = 100$

Une des 15 solutions est : $9 - 8 + 7 + 65 - 4 + 32 - 1 = 100$

On dispose des trois fonctions et de la liste suivantes :

```
def addition(a,b):
    return a+b
def soustraction(a,b):
    return a-b
def concatenation(a,b):
    return (int(str(a)+str(b)))
Loperations=[addition,soustraction,concatenation]
```

En Python, les fonctions sont des objets comme les autres que l'on peut donc manipuler.

Exemple :

```
for operation in Loperations:
    print(operation(4, 3))
```

va successivement appliquer les opérations (qui sont des fonctions) de la liste `Loperations` et donc afficher :

7
1
43

Dans tout le problème, Ndésigne un entier naturel non nul.

On appelera **liste d'indices d'opérations** toutes listes composées de 0 , 1 ou 2.

On appelera **calcul** associé à la liste d'entiers naturels Lnb de longueur N et à la liste d'indices d'opérations Mop de longueur $N-1$, l'entier $calc$ (unique) obtenu en intercalant les opérations de `Loperations` associées aux éléments de Mop entre les éléments de Lnb.

0 représentant toujours l'addition. 1 représentant toujours la soustraction.

2 représentant toujours la concaténation.

Exemple :

Si $Lnb = [1,2,3,4,5,6,7,8,9]$ et $Mop = [2,2,0,1,0,2,1,2]$ alors

$calc = 123 + 4 - 5 + 67 - 89 = 100$

On appelera **représentation décimale** Rd de la liste d'indices d'opérations Mop l'entier correspondant à l'écriture en base 3 de Mop.

L'écriture est dans l'ordre des puissances croissantes ; on commence par les poids faibles.

Exemple :

Si $Mop = [2,2,0,1,0,1]$ alors

$Rd = 2\times3^0 + 2\times3^1 + 0\times3^2 + 1\times3^3 + 0\times3^4 + 1\times3^5 = 2 + 6 + 27 + 243 = 278$

1. (a) Quel est le calcul associé à $[1,2,3,4,5,6,7,8,9]$ et à $[0,2,1,0,0,0,2,1]$?

   (b) On a : $1 + 23 - 4 + 5 + 67 + 8 = 100$.
   En déduire une liste d'indices d'opérations $M1$ telle que le calcul associé à $[1,2,3,4,5,6,7,8]$ et $M1$ soit égal à 100.

   (c) Déterminer une liste d'indices d'opérations $M2$, différente de $M1$, telle que le calcul associé à $[1,2,3,4,5,6,7,8]$ et $M2$ soit égal à 100.

   (d) Ecrire la liste d'indices d'opérations $M3$ de longueur 8 dont la représentation décimale est 497.
   Remarque : $3^4 = 81$, $3^5 = 243$ et $3^6 = 729$.

2.(a) On souhaite obtenir une fonction `genLnb` prenant en entrée un entier N et retournant une liste de chiffres (appartenant à $[|1,9|]$) de longueur N créée aléatoirement.
Exemple :
`genLnb(12)` retourne $[3,5,8,9,6,4,1,8,9,7,7,2]$.
On indique, que `rd.randint`(a, b), pour deux entiers positifs a et b, renvoie un entier dans $[|a,b|]$ choisi aléatoirement de manière uniforme.
Recopier et compléter :

```
import random as rd
def genLnb(N):
    Lnb=[]
    while #Acompleter :
        x=rd.randint(1,9)
        #Acompleter
    return Lnb
```

(b) En modifiant un peu la fonction précédente, écrire de même une fonction `genMop` prenant en entrée un entier N et retournant une liste d'indices d'opérations de longueur $N-1$ créée aléatoirement.
Exemple : `genMop`(10) retourne $[2,1,0,0,2,1,0,0,1]$.

3. On dispose d'une fonction `conversionBase` prenant en entrée un entier X, un entier $b>1$ et un entier *long* tel que $0 \leqslant X < b^{long}$ et retournant une liste de longueur *long* représentant l'écriture en base b dans l'ordre des puissances croissantes de l'entier X.
Exemple :
`conversionBase`(145,7,8) retourne [5, 6, 2, 0, 0, 0, 0, 0]

```
def conversionBase(X, b, long):
    res = []
    reste = X
    i = 0
    while reste > 0:
        res.append(reste % b)
        reste = reste // b
        i += 1
    return res + [0] * (long - len(res))
```

(a) Justifier la *terminaison* de la fonction `conversionBase`.

(b) On souhaite justifier la *correction* de la fonction `conversionBase`, c'est-à-dire que `conversionBase` permet bien de réaliser la tâche voulue.
On écrit la décomposition en base b de X :
Il existe $(x_0, x_1, ..., x_p) \in [|0, b-1|]^{p+1}$ avec $x_p \neq 0$ tel que : $X = \sum_{j=0}^{p} x_j b^j$
Compléter et justifier l'invariant de boucle et montrer la correction de `conversionBase` :
" Après i tours de boucle : res = [...] et reste $= \sum_{k=...}^{...} x_k b^{...}$ "

(c) Déterminer en fonction de X et de b le nombre de tours de boucle effectués lors de l'exécution de `conversionBase`(X,b).
En déduire la *complexité temporelle* de `conversionBase` en fonction de X et b.

4.(a) Ecrire une fonction `transformation` prenant en entrée une liste d'entiers naturels *Lnb* de longueur N et une liste d'indices d'opérations *Mop* de longueur $N-1$ et retournant une liste d'entiers naturels *LnbT* de longueur $N-c$ et une liste d'indices d'opérations *MopT* de longueur $N-c-1$ représentant les deux listes de départ après avoir effectué les c concaténations imposées par *Mop*.
Exemple : `transformation`$([1,2,3,4,5,6,7],[2,2,1,2,0,2])$
retourne $([123,45,67],[1,0])$.
Remarque : On pourra utiliser la fonction `concatenation` du préambule.
Donner la *complexité temporelle* `transformation` en fonction de N.

(b) En utilisant la fonction précédente, écrire une fonction `calcul` prenant en entrée une liste d'entiers naturels *Lnb* de longueur N et une liste d'indices d'opérations *Mop* de longueur $N-1$ et retournant le calcul associé aux listes *Lnb* et *Mop*.
Exemple :
`calcul`$([1,2,3,4,5,6,7],[2,2,1,2,0,2]$) retourne 145 (car $123-45+67=145$).
Donner la *complexité temporelle* en fonction de N.

5. On souhaite écrire une fonction `somEgal` prenant en entrée une liste d'entiers naturels *Lnb* de longueur au moins 1 et un entier *resultat* et retournant la liste contenant les représentations décimales de toutes les listes d'indices d'opérations *Mop* telles que le calcul associé à *Lnb* et à *Mop* soit égal à *resultat*.
Exemples :
`somEgal`([1,2,3,4,5],51) retourne [20, 54].
`somEgal`([1,2,3,4,5],50) retourne [].
`somEgal`$([1,2,3,4,5,6,7,8,9],100)$ retourne
$[177,497,1260,1467,2555,2816,3660,4469,4700,5012,5624]$
Pour ce dernier exemple, on peut effectuer une vérification :
177 est la représentation décimale de la liste d'indices d'opérations $[0,2,1,0,2,0,0,0,0]$.
le calcul associé à $[1,2,3,4,5,6,7,8,9]$ et à $[0,2,1,0,2,0,0,0,0]$ est bien :
$1+23-4+56+7+8+9=100$

(a) Ecrire la fonction `somEgal`.
Indication :
Pour les listes *Mop* candidates, on utilisera `conversionBase`(x,3,$N-1$) avec N =? et pour x décrivant ...

(b) Donner la *complexité temporelle* en fonction de N la longueur de *Lnb*.

**********FIN MPSI3**********
Avec l'ombre de Lewis Carroll [1] planant de nouveau pas très loin...

---

6. En utilisant, la fonction `transformation`, écrire une fonction `listToChaine` prenant en entrée une liste d'entiers naturels *Lnb* de longueur N et une liste d'indices d'opérations *Mop* de longueur $N-1$ et retournant une chaine de caractères *ch* correspondant à l'écriture du calcul associé aux listes *Lnb* et *Mop*.
Exemple :
`listToChaine`$([1,2,3,4,5,6,7,8,9],[2,2,1,1,1,1,0,1])$ retourne '123-4-5-6-7+8-9'.

7. Ecrire une fonction `presentation` prenant en entrée une liste d'entiers naturels *Lnb* et un entier *resultat* et retournant l'ensemble des expresions algébriques permettant d'obtenir *resultat* en intercalant des opérations de la liste `Loperations` entre les éléments de la liste de nombres *Lnb*.
Exemple : `presentation`$([1,2,3,4,5,6,7,8,9],100)$ retourne :

```
1 + 23 - 4 + 56 + 7 + 8 + 9 ; 12 + 3 - 4 + 5 + 67 + 8 + 9
1 + 2 + 34 - 5 + 67 - 8 + 9 ; 1 + 2 + 3 - 4 + 5 + 6 + 78 + 9
123 - 4 - 5 - 6 - 7 + 8 - 9 ; 123 + 45 - 67 + 8 - 9
1 + 23 - 4 + 5 + 6 + 78 - 9 ; 12 - 3 - 4 + 5 - 6 + 7 + 89
12 + 3 + 4 + 5 - 6 - 7 + 89 ; 123 - 45 - 67 + 89
123 + 4 - 5 + 67 - 89
```

Remarque : On **utilisera** les fonctions `somEgal`, `listToChaine` et `conversionBase`.

---

1. 
Lewis Carroll (Daresbury, 1832 - 1898, Guildford) pseudonyme de Charles Lutwidge Dodgson, était à la fois un célèbre romancier et un professeur de logique à Christ Church College (université de Oxford). Il est principalement connu pour son roman **Les Aventures d'Alice au pays des merveilles** (remember your last test !) et son poème La Chasse au Snark. Lewis Carroll a travaillé dans les domaines de la logique mathématique, des probabilités et des `mathématiques récréatives`, produisant près d'une douzaine de livres sous son vrai nom. *la méthode de Dodgson* est utilisée pour l'étude des élections

8. On considère la fonction `obtention`.

```
def obtention(chiffre,resultat):
    if resultat % chiffre !=0:
        return False
    else:
        L , trouve= [] , False
        while not trouve:
            L.append(chiffre)
            solution=somEgal(L,resultat)
            trouve = ( len(solution) >0 )
            if trouve:
                X=solution[0]
                return affichage(L,convert(X,3,len(L)-1))
```

(a) Sans utiliser la fonction `len`, proposer une écriture équivalente pour la ligne :
trouve = (len(solution) >0).

(b) Que retourne `obtention`$(8, 1001)$?

(c) Que retourne `obtention`$(8, 816)$?

(d) Décrire sommairement ce que permet d'obtenir `obtention` en fonction de *chiffre* et de *resultat*.

9.(a) On veut disposer d'une fonction `creationFichier` prenant en entrée une chaine de caractères *nomFichier* et deux entiers $Nblignes \geqslant 1$ et $Nmax \geqslant 2$ et retournant un fichier nommé *nomFichier* dont chacune des *Nblignes* lignes est une chaine de caratères contenant une expression algébrique (sans espace) correspondant au calcul d'une liste de chiffres appartenant à $[|1, 9|]$ de longueur au plus $Nmax$.

Exemple : `creationFichier`("FICHIER1",4,20) peut permettre de créer.

```
7 + 564 - 5 + 7 + 862 - 4 - 6 - 61335          #longueur=16
2 + 33 + 7 + 55                                #longueur=6
67 + 1 + 7 - 2 + 2 + 84 + 927                  #longueur=11
4714                                           #longueur=4
```

Recopier et compléter la fonction suivante.

```
def creationFichier(nomFichier,Nblignes,Nmax):
    of=open(nomFichier,#Acompleter)
    for k in #Acompleter:
        n=rd.randint(2,#Acompleter)
        Lnb=#Acompleter(n)
        Mop=genMop(#Acompleter)
        ligne=#Acompleter
        of.#Acompleter# (ligne+'\n')
    of.#Acompleter
```

(b) Ecrire une fonction `recherche` prenant en entrée un fichier *fich* du type précédente et un entier *resultat* et retournant l'ensemble des lignes de *fich* telles que le calcul de l'expresion algébrique associée soit égal à *resultat*.

Exemple : Avec *fich*2 valant :

```
56+98-7+86-432-6
44-8+1
2+19-48-77+8+63
1+9-8-7+42
57-1+48
```

`recherche`(*fich*2,37) retourne :
$44 - 8 + 1$
$1 + 9 - 8 - 7 + 42$.