```python
# Correction_DS2_Info_MPSI_2023_2024.py

##Exercice - d'après CCP MP
#Q1.
def estPremier(n):
    '''Un nombre premier est un entier positif
ayant exactement deux diviseurs positifs'''
    nb_div = 0
    for k in range(1,n+1):
        if n%k == 0:
            nb_div += 1
    if nb_div == 2:
        return True
    else:
        return False

#Autre version
def estPremier(n):
    if n >= 2:
        cpt = 0  #compte le nbre de diviseurs de n
supérieurs ou égaux à 2
        d = 2
        while d**2 <= n:
            if n%d == 0:
                cpt += 1
            d += 1
        if cpt == 0:
            return True
    return False

#Q2.
def liste_premier(n):
    L = []  #initialisation de la liste des
nombres premiers
    for k in range(2,n+1):
        if estPremier(k):
            L.append(k)
    return L

#Q3.
```

```
def valuation_p_adique(n,p):
    puiss = 0
    quotient = n
    while quotient % p ==0:
        puiss += 1
        quotient = quotient // p
    return puiss

#Q4.
def val(n, p):
    if n%p !=0:
        return 0
    else:
        return 1+val(n//p, p)

#Q5.
def decomposition_facteurs_premiers(n):
    rep = []
    L = liste_premier(n)  #liste des nombres
premiers inférieurs ou égaux à n
    for p in L:
        if n%p == 0: #si p divise n
            vp = val(n,p)
            rep.append((p,vp))
    return rep

##Problème : élections présidentielles
#Q1.
'''l'instruction est n = len(C)'''

#Q2a.
def appartient(L,x,i0):
    n = len(L)
    for k in range(i0,n):  #k décrit les indices
de i0 à n-1
        if L[k] == x:
            return True
    return False   #si on est là c'est que x n'a
pas été trouvé
```

```
#Q2b.
def tousDiff(L):
    n = len(L)
    for i0 in range(n):  #i0 décrit les indices de
0 à n-1
         #on considère L[i0] et on regarde s'il
est présent dans C
         #à un indice supérieur ou égal à i0+1
        x = L[i0]
        if appartient(L,x,i0+1):
            return False
    return True

#Q3a.
def nbOcc(V,s):
    cpt = 0  #compte le nombre d'apparitions de s
dans V
    for x in V:
        if s==x:
            cpt += 1
    return cpt

#Q3b.
def makeR(C,V):
    rep = []  #liste des résultats
    for cand in C:  #cand décrit les prénoms des
différents candidats
        nb = nbOcc(V,cand)
        rep.append(nb)
    return rep

#Q4a.
def nomToNum(C,s):
    n = len(C)
    for i in range(n):  #i décrit les indices de C
de 0 à n-1
        if C[i]==s:
            return i
    return None   #Cette ligne peut être ommise
auquel cas la fonction renvoie None si s n'a pas
```

```
été trouvé

#Q4b.
def makeRbis(C,V):
    n = len(C)  #nbre de candidats
    rep = [0]*n  #on initialise la liste des
résultats à [0,...,0] (n termes)
    for vote in V: #pour chaque vote
        i = nomToNum(C,vote) #on récupère le
numéro du candidat correspondant
        if i !=None:  #si le vote n'est pas nul ou
blanc
            rep[i] += 1 #on ajoute 1 au bon
endroit dans la liste des résultats
    return rep

#Q5a.
def nbVotes(R):
    #il s'agit simplement de calculer la somme des
éléments de R
    nb = 0
    for x in R:
        nb += x
    return nb

#Q5b.
def majAbs(R):
    nb = nbVotes(R)
    return int(nb/2) + 1

#Q6a.
def vainqueurTour1(R):
    maj = majAbs(R) #majorité absolue
    for i in range(len(R)):  #i décrit les indices
de R de 0 à n-1
        if R[i] >= maj:
            return i
    return None
```

```python
#Q6b.
def deuxMeilleurs(R):
    #initialisation de i1 et i2 selon les
positions relatives de R0 et R1
    if R[0] >= R[1]: #>= pour avoir i1<i2 en cas
d'égalité
        i1,i2 = 0, 1
    else:
        i1, i2 = 1, 0
    for i in range(2,len(R)): #i varie de 2 à
len(R)-1
        x = R[i]
        #mise à jour éventuelle des valeurs de i1
et i2 selon la valeur de R[i]
        if x > R[i1]: #nouveaux max : R[i] puis
R[i1]
            i2 = i1
            i1 = i
        elif R[i] > R[i2]: # on aura forcément ici
R[i1] >= R[i]
        #nouveaux max : R[i1] puis R[i], inégalité
> pour que i2 soit le plus petit possible
            i2 = i
    return (i1,i2)

#Q7a.
def valAbs(a):
    if a >= 0:
        return a
    return -a
```

```python
#Q7b.
def vainqueurTour2(R, i1, i2):
```

```
    nv_c1 = 0  #nbre de votes pour le candidat 1
    nv_c2 = 0  #nbre de votes pour le candidat 2
    for i in range(len(R)):
        if valAbs(i-i1) <= valAbs(i-i2):
            nv_c1 += 1
        else:
            nv_c2 += 1
    if nv_c1 > nv_c2:
        return i1
    elif nv_c2 > nv_c1:
        return i2
    elif i1 < i2:
    #cas d'égalité entre les 2 candidats, on
renvoie le candidat avec le plus petit numéro
        return i1
    else: return i2

#Q8.
def vainqueurElection(C,V):
    if not tousDiff(C):
        return  #on renvoie None
    R = makeR(C,V)  #liste des résultats
    ind_V1 = vainqueurTour1(R)  #indice du
vainqueur du 1er tour
    if ind_V1 != None: #cas où il y a un vainqueur
au 1er tour
        return C[ind_V1]
    else: #cas où il y aura un 2ème tour
        i1,i2 = deuxMeilleurs(R)
        ind_V = vainqueurTour2(R,i1,i2) #indice du
vainqueur
        return C[ind_V]

#Q9.
''' il faut importer le module random'''
import random



#Q10.
```

```
"""Fonction f :
m étant un entier (type int),
renvoie une chaine de caratère (type str) composée
de m caractères choisis
aléatoirement entre les lettres a et z"""

"""Fonction g :
n étant un entier (type int),
renvoie une liste (type list) de n chaines de
caractères aléatoires
contenant entre 4 et 10 caractères chacune (nbre
de caractères aléatoires),
toutes différentes"""

#Q11
def listeAlea(m, s):
    a1 = random.randint(0,s)  #entier entre 0 et s
    A = [a1]  #initialisation de la liste
aléatoire
    reste = s - a1  #stocke s - somme des ai
    for k in range(2,m):  #pour k allant de 2 à
m-1
        ak = random.randint(0,reste)  #entier
aléatoire suivant
        A.append(ak)
        reste -= ak  #mise à jour de reste
    A.append(reste)  #ajout de am, qui vaut reste
    return A

#Q12a.
def tableau(m,s):
    liste = [0]*(m+s-1)  #liste initialisée avec
(m+s-1) zéros
    nb_1 = 0  #nombre de 1 présents dans la liste
    while nb_1 < m-1:
        j = random.randint(0,m+s-1)
        if liste[j] == 0:
            liste[j] = 1
            nb_1 += 1
    return liste
```

```
#Q12b.
def nbCase(L,i):
    """Etant donnée une liste L (de 0 et de 1) et
un entier i,
    renvoie le nbre de cases nulles depuis la case
i (incluse)
    jusqu'à la prochaine case valant 1 et l'indice
du 1 suivant"""
    cpt = 0  #nbre de cases
    j = i  #indice de la case en cours, initialisé
à i
    while j<len(L) and L[j] == 0:
        cpt += 1
        j += 1
    return cpt, j

def listeAlea2(m,s):
    T = tableau(m,s)
    ak,ik = nbCase(T,0)  #valeurs de a1 et i1
(indice du 1er 1)
    A = [ak]  #initialisation de la liste
aléatoire à [a1]
    for k in range(2,m): #k varie de 2 à m-1
        ak,ik = nbCase(T,ik + 1)  #valeurs de ak
et ik (indice du kème 1)
        A.append(ak)
    am = s+m-1-(ik+1)  #nbre de cases après le
dernier 1
    A.append(am)
    return A
#Q13.
def genererR(n,s):
    v1,v2,m = listeAlea2(3,s)
    R = listeAlea2(n,m)
    return R, v1, v2

#Q14.
def melange(L):
    n = len(L)
```

```
    for i in range(n-1,0,-1):  #i décrit les
entiers de n-1 à 1 (avec un pas de -1)
        j = rd.randint(0,i)
        L[i],L[j]=L[j],L[i]
    #pas de return : la liste L est directement
modifiée par la fonction

#Q15.
def genererV(C, R, v1, v2):
    V = [""]*v1 + [f(11)]*v2  #initialisation de V
avec les votes blancs et nuls
    #puis ajout du bon nombre de votes pour chaque
candidat
    for i in range(len(R)):
        cand = C[i]
        nb = R[i]
        V += nb*[cand]
    melange(V)  #mélange de V
    return V

#Q16.
def test(n, s):
    C = g(n)  #liste de candidats
    R,v1,v2 = genererR(n,s)  #liste de résultats,
de votes blancs et nuls
    V = genererV(C, R, v1, v2)  #liste des votes
correspondante
    R1 = makeR(C,V)
    R2 = makeRbis(C, V)
    if R == R1 and R == R2:
        return True
    return False
```