INFORMATIQUE : CORRECTION FORMATIF N°2 - BCPST 1

I. (24 points) Comparaison de deux séquences d'ADN.

```
ARNm = 'CCAGCCAUGUGCUACAUCCACCAAGUACCCCCUUGGUAGUUUUUA'

codon = [
'UUU', 'UUC', 'UUA', 'UUG', 'CUU', 'CUC', 'CUA', 'CUG',
'AUU', 'AUC', 'AUA', 'AUG', 'GUU', 'GUC', 'GUA', 'GUG',
'UCU', 'UCC', 'UCA', 'UCG', 'CCU', 'CCC', 'CCA', 'CCG',
'ACU', 'ACC', 'ACA', 'ACG', 'GCU', 'GCC', 'GCA', 'GCG',
'UAU', 'UAC', 'UAA', 'UAG', 'CAU', 'CAC', 'CAA', 'CAG',
'AAU', 'AAC', 'AAA', 'AAG', 'GAU', 'GAC', 'GAA', 'GAG',
'UGU', 'UGC', 'UGA', 'UGG', 'CGU', 'CGC', 'CGA', 'CGG',
'AGU', 'AGC', 'AGA', 'AGG', 'GGU', 'GGC', 'GGA', 'GGG'
]

acide = [
'Phe', 'Phe', 'Leu', 'Leu', 'Leu', 'Leu', 'Leu','Leu',
'Ile', 'Ile', 'Ile', 'Met', 'Val', 'Val', 'Val', 'Val',
'Ser', 'Ser', 'Ser', 'Ser', 'Pro', 'Pro', 'Pro', 'Pro',
'Thr', 'Thr', 'Thr', 'Thr', 'Ala', 'Ala', 'Ala', 'Ala',
'Tyr', 'Tyr', 'Stop', 'Stop', 'His', 'His', 'Gin', 'Gin',
'Asn', 'Asn', 'Lys', 'Lys', 'Asp', 'Asp', 'Glu', 'Glu',
'Cys', 'Cys', 'Stop', 'Trp', 'Arg', 'Arg', 'Arg', 'Arg',
'Ser', 'Ser', 'Arg', 'Arg', 'Gly', 'Gly', 'Gly', 'Gly'
]
```

(Copie du fichier : 1 pt ; restitution correcte du devoir : 1 pt)

1°) (3 points)

```
#Détermination de l'index du codon d'initiation
#version avec la fonction index
ind_cod_ini = ARNm.index('AUG')

#Détermination de l'index du codon d'initiation
#version sans la fonction index
for i in range(len(ARNm) - 2):
    if ARNm[i:i + 3] == 'AUG':
        ind_cod_ini = i
print("Index du codon d'initiation =", ind_cod_ini)
```

2°) (5 points)

```
#Détermination de l'index du codon stop
#version avec la fonction index
ind_cod_stop1 = ind_cod_stop2 = ind_cod_stop3 = len(ARNm)
if 'UAA' in ARNm:
    if (ARNm.index('UAA') - ind_cod_ini)%3 == 0:
        ind_cod_stop1 = ARNm.index('UAA')
if 'UAG' in ARNm:
    if (ARNm.index('UAG') - ind_cod_ini)%3 == 0:
        ind_cod_stop2 = ARNm.index('UAG')
if 'UGA' in ARNm:
```

```
    if (ARNm.index('UGA') - ind_cod_ini)%3 == 0:
        ind_cod_stop3 = ARNm.index('UGA')
ind_cod_stop = min(ind_cod_stop1, ind_cod_stop2, ind_cod_stop3)

#Détermination de l'index du codon stop
#version sans la fonction index
for i in range(ind_cod_ini, len(ARNm) - 3, 3):
    if ARNm[i:i + 3] == 'UAA' or ARNm[i:i + 3] == 'UAG' or ARNm[i:i + 3] == 'UGA':
        ind_cod_stop = i

print("Index du codon stop =", ind_cod_stop)
```

3°) (6 points)

```
def anti_codon(ARNm1):
    """Détermine la chaine d'anti-codon"""
    base = 'ACGU'
    anti = 'UGCA'
    antcod = ''
    for el in ARNm1:
        antcod += anti[base.index(el)]
    return antcod
```

4°) (6 points)

```
def acide_ami(ARNm1, ind_cod_ini1, ind_cod_stop1):
    """Traduit le codon en acide aminé"""
    proteine = ''
    for i in range(ind_cod_ini1, ind_cod_stop1, 3):
        cod = ARNm1[i:i + 3]
        proteine += acide[codon.index(cod)] + ' '
    return proteine
```

5°) (4 points)

```
antcod = anti_codon(ARNm) #(0,5 pt)
print(antcod[ind_cod_ini:ind_cod_stop]) #(0,5 pt)
print ((ind_cod_stop - ind_cod_ini) * '|', end = '') (1 pt)
print ()
print(ARNm [ind_cod_ini:ind_cod_stop]) #(0,5 pt)
print('\nProtéine =', acide_ami(ARNm, ind_cod_ini,ind_cod_stop)) #(1 pt)
```

(Affichage avec le codon d'initiation et le codon stop exclus : 0,5 pt)

```
Index du codon d'initiation = 6
Index du codon stop = 36
UACACGAUGUAGGUGGUUCAUGGGGGAACC
||||||||||||||||||||||||||||
AUGUGCUACAUCCACCAAGUACCCCCUUGG
Protéine = Met Cys Tyr Ile His Gin Val Pro Pro Trp
```

II. (16 points) Introduction au jeu "Le compte est bon".

1°) (1 point)

```
from random import randint, sample
resultat = randint(100, 999)
```

2°) (3 points)

```
def ch_plaques():
    """Renvoie la liste des 6 plaques"""
    nb = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100]
    return [nb[randint(0, len(nb) - 1)] for i in range(6)]
ou
    return sample(nb, 6)
```

3°) (Bonus : 4 points)

```
def paires(li_plaques):
    """Renvoie la liste des 15 paires de plaques possibles"""
    li_paires = []
    for i in range(len(li_plaques) - 1):
        for j in range(i + 1, len(li_plaques)):
            li_paires += [li_plaques[i], li_plaques[j]]
    return li_paires
```

4°) (5 points)

```
def result_ope(a,b):
    """Renvoie la liste des résultats possibles"""
    li_result = []
    li_result.append(a + b)
    li_result.append(abs(a - b))
    li_result.append(a * b)
    if b > a and b%a == 0 and (a != 0 or a != 1):
        li_result.append(int(b / a))
    elif a > b and a%b == 0 and (b != 0 or b != 1):
        li_result.append(int(a / b))
    else:
        li_result.append(0)
    return li_result
```

5°) (5 points)

```
print('Valeur à calculer =', resultat)
li_plaques = ch_plaques()
print('Plaques =', end = ' ')
for i in li_plaques:
    print(i, end = '  ')
print('\nPaire de plaques =')
li_paires = paires(li_plaques)
for i in range(0, len(li_paires) - 1, 2):
    print(li_paires[i], li_paires[i + 1], end = ' | ')

li_paires_ope = result_ope(li_paires[0], li_paires[1])
print('\nPlaque 1 =', li_paires[0], 'Plaque 2 =', li_paires[1])
print('Résultats possibles =', li_paires_ope)
```