Exercice 1 (6 points)

Cet exercice porte sur la programmation Python et la cryptographie.

Le chiffrement Playfair, popularisé par Lord Playfair et utilisé par l'armée britannique durant les guerres du XXème siècle, est basé sur le chiffrement de paires de lettres (appelées **digrammes**).

Partie A : la clef de chiffrement

Ce chiffrement utilise un tableau de 5×5 lettres contenant un mot clef. On remplit le tableau avec les lettres du mot clef (sans doublons), puis on le complète avec les lettres restantes de l'alphabet (sans la lettre W) dans leur ordre alphabétique. Une lettre n'apparait qu'une seule fois dans le tableau.

Par exemple, si on choisit comme clef le mot **PLAYFAIR**, le carré de chiffrement obtenu est le suivant :

P	L	A	Y	F
I	R	B	C	D
E	G	H	J	K
M	N	O	Q	S
T	U	V	X	Z

Figure 1. Carré de chiffrement obtenu avec le mot clef PLAYFAIR

On commence par les lettres de la clef (cases blanches) sans les doublons (ici le A) puis on complète le tableau (cases grisées) avec les lettres restantes de l'alphabet, dans l'ordre alphabétique.

1. Donner le carré de chiffrement si la clef est `EPREUVEDENSI`.

On donne ci-dessous le code Python de la fonction `creer_liste_clef` qui prend en paramètre la clef de chiffrement et renvoie une liste contenant 25 lettres ordonnées de la façon suivante : d'abord les lettres de la clef choisie (sans doublon) puis les lettres de l'alphabet restantes (classées par ordre alphabétique).

```
 1 def creer_liste_clef(clef):
 2     """
 3     hypothèse : la clef ne contient pas la lettre W
 4     """
 5     deja_utilises = []
 6     # alphabet sans la lettre W:
 7     alphabet = 'ABCDEFGHIJKLMNOPQRSTUVXYZ'
 8     for i in range(len(clef)):
 9         if not (clef[i] in deja_utilises):
10             deja_utilises.append(clef[i])
11     for lettre in alphabet:
12         if not lettre in deja_utilises:
13             deja_utilises.append(lettre)
14     return deja_utilises
```

Exemple :

```
creer_liste_clef('PLAYFAIR')
>>> ['P', 'L', 'A', 'Y', 'F', 'I', 'R', 'B', 'C', 'D', 'E',
'G', 'H', 'J', 'K', 'M', 'N', 'O', 'Q', 'S', 'T', 'U', 'V',
'X', 'Z']
```

2. Donner l'assertion à insérer en début de la fonction `creer_liste_clef` afin de s'assurer que l'hypothèse sur la clef soit respectée.

On donne ci-dessous le code incomplet de la fonction `creer_carre` qui prend en paramètre la liste créée par la fonction `creer_liste_clef` et renvoie le carré de chiffrement.

```
1 def creer_carre(liste_clef):
2     carre = [[0 for i in range(5)] for j in range(5)]
3     for i in range(25):
4         carre[...][...] = liste_clef[i]
5     return carre
```

Exemple :

```
creer_carre(creer_liste_clef('PLAYFAIR'))
>>> [['P', 'L', 'A', 'Y', 'F'], ['I', 'R', 'B', 'C', 'D'],
['E', 'G', 'H', 'J', 'K'], ['M', 'N', 'O', 'Q', 'S'], ['T',
'U', 'V', 'X', 'Z']]
```

3. Recopier et compléter la ligne 4 du code de cette fonction `creer_carre`, en utilisant les opérateurs `%` (reste de la division entière) et `//` (division entière).

Partie B : chiffrer un message

Le chiffrement d'un message se fait en deux étapes :

- 1ère étape : on découpe le message en digrammes (paires de lettres) ;
- 2ème étape : on chiffre chacun des digrammes avec le carré de chiffrement.

Pour découper le message en digrammes, on prend les lettres deux par deux en tenant compte de deux cas particuliers :

- si les deux lettres sont identiques, on ajoute un `'X'` après la première lettre et on poursuit le découpage deux à deux à partir de la deuxième lettre ;
- s'il ne reste qu'une seule lettre, on forme une dernière paire en lui ajoutant la lettre un `'X'`.

Par exemple :

le découpage de `'BACCALAUREAT'` donnera `'BA', 'CX', 'CA', 'LA', 'UR', 'EA', 'TX'`

Le chiffrement d'un message se fait ensuite en chiffrant chaque digramme (paire de lettres), de la manière suivante :

- si les lettres du digramme se trouvent sur la même ligne du carré de chiffrement, il faut les remplacer par celles se trouvant immédiatement à leur droite (en bouclant sur la gauche si le bord est atteint) ;
- si les lettres apparaissent sur la même colonne du carré de chiffrement, les remplacer par celles qui sont juste en dessous (en bouclant par le haut si le bas de la table est atteint) ;
- sinon, remplacer les lettres par celles se trouvant sur la même ligne du carré de chiffrement, mais dans le coin opposé du rectangle défini par la paire originale.

Par exemple, si le message est `'VIVELANSI'`, les digrammes sont : `'VI', 'VE', 'LA', 'NS', 'IX'` et leurs codages avec la clef `PLAYFAIR` sont :

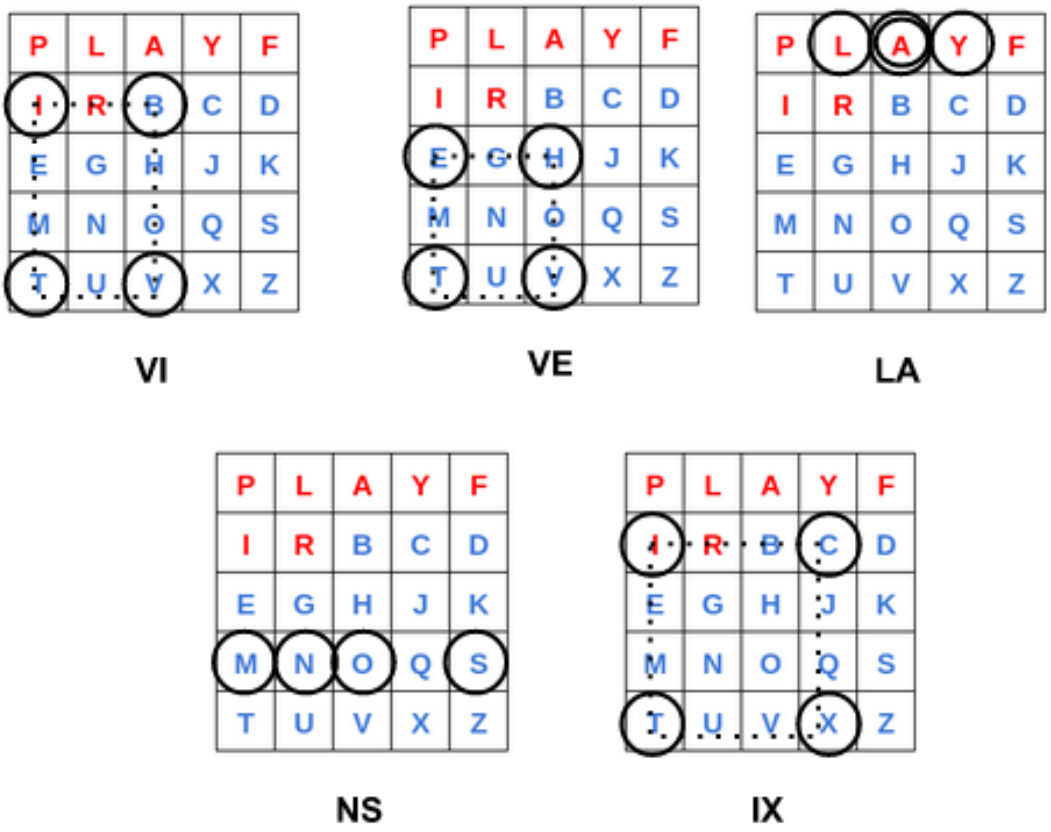

Figure 2. Chiffrement de quelques digrammes

digramme	VI	VE	LA	NS	IX
chiffré	TB	TH	AY	OM	CT

On donne ci-dessous le code incomplet de la fonction `couper_en_digrammes` qui prend en paramètre une chaine de caractères et renvoie la liste des digrammes la constituant :

```
def couper_en_digrammes(message):
    digrammes = []
    i = 0
    while i < len(message) - 1:
        if message[i] == message[i+1]:
            digrammes.append(message[i] + 'X')
            i = i + 1
        else:
            ...
            i = i + 2
    if i == len(message) - 1: #il reste une lettre isolée
```

```
12          digrammes.append(message[i] + 'X')
13      return digrammes
```

4. Donner le code de la ligne 9 manquante de cette fonction `couper_en_digrammes`.

5. Donner le résultat de l'appel `couper_en_digrammes('BONJOUR')`.

6. Donner le chiffrement du message `BONJOUR` avec le carré de chiffrement `PLAYFAIR` donné en Figure 1.

7. Donner le code Python de la fonction `ligne_colonne` qui prend en paramètres une lettre et le carré de chiffrement créé par la fonction `creer_carre`, et qui renvoie les coordonnées de la lettre dans le carré de chiffrement.

   Exemple (avec le carré de chiffrement de la Figure 1) :

   ```
   ligne_colonne('A', carre)
   >>> (0, 2)

   ligne_colonne('N', carre)
   >>> (3, 1)
   ```

8. Donner le code Python de la fonction `sur_la_meme_ligne` qui prend en paramètres un digramme et le carré de chiffrement créé par la fonction `creer_carre`, et qui renvoie `True` si les deux lettres du digramme sont sur la même ligne, ou `False` sinon.

   Exemple (avec le carré de chiffrement de la Figure 1):

   ```
   sur_la_meme_ligne('RT', carre)
   >>> False

   sur_la_meme_ligne('PL', carre)
   >>> True
   ```

On dispose pour la suite de la fonction `sur_la_meme_colonne`, similaire à `sur_la_meme_ligne` mais en colonne.

Voici le code incomplet de la fonction `chiffrer_digramme` qui prend en paramètres le carré de chiffrement et un digramme, et qui renvoie le digramme chiffré correspondant :

```
1 def chiffrer_digramme(digramme, carre):
2     lettre1 = digramme[0]
3     lettre2 = digramme[1]
4     i1, j1 = ligne_colonne(lettre1, carre)
5     i2, j2 = ligne_colonne(lettre2, carre)
6     if sur_la_meme_ligne(digramme, carre):
7         digramme_chiffre = carre[i1][(j1 + 1)%5] +
```

```
carre[i2][(j2 + 1)%5]
 8      elif sur_la_meme_colonne(digramme, carre):
 9          digramme_chiffre = ...
10      else:
11          digramme_chiffre = ...
12      return digramme_chiffre
```

9. Donner le code complet des lignes 9 et 11 de cette fonction `chiffrer_digramme`.

10. Écrire le code python de la fonction `chiffrer_playfair` qui prend en paramètres deux chaînes de caractères `message` et `clef` correspondant au message à chiffrer et au mot-clef choisi, et qui renvoie le message chiffré, en utilisant les fonctions déjà écrites précédemment.

    Exemple :

    ```
    chiffrer_playfair('VIVELANSI', 'PLAYFAIR')
    >>> 'TBTHAYOMCT'
    ```