```python
# ================================================================================
# DM2 - Version Exercices
# ================================================================================

#%% Exercice 1
def positions(liste,elt):
    liste_pos=[]
    for i in range(len(liste)):
        if liste[i]==elt:
            liste_pos.append(i)
    return liste_pos

#%% Exercice 2
def renverse(chaine):
    chaine_inv=''
    n=len(chaine)
    for i in range(n):
        chaine_inv=chaine_inv+chaine[n-1-i]
    return chaine_inv


#%% Exercice 3
def max_pair(liste):
    maxi=-float('inf')
    for x in liste:
        if x%2==0 and x>maxi:
            maxi=x
    if maxi==-float('inf'):
        return None
    else:
        return maxi

#%% Exercice 4
def deuxieme(liste):
    #on cherche d'abord le plus grand
    maxi=-float('inf')
    for x in liste:
        if x>maxi:
            maxi=x

    #recherche du deuxieme plus grand
    deux=-float('inf')
    for x in liste:
        if x!=maxi and x>deux:
            deux=x

    if deux==-float('inf'):
        return None
    else:
        return deux

#%% Exercice 5
def doublons(liste):
    liste2=[]
    for x in liste:
        if x not in liste2:
            liste2.append(x)
    return liste2
#Cette fonction est de complexité quadratique car on a:
    # une boucle qui parcourt la liste de départ
    #et à l'intérieur le test x not in liste2 qui parcourt liste2

#On améliore la complexité grâce à un dictionnaire
#La complexité devient linéaire car la recherche d'une clef dans un dictionnaire
# est de complexité constante
def doublons2(liste):
    liste2=[]
    d={}
    for x in liste:
        if x not in d:
            liste2.append(x)
            d[x]='present'
```

```
    return liste2

#%% Exercice 6
def croissante(liste):
    n=len(liste)
    for i in range(n-1):
        if liste[i+1]<=liste[i]:
            return False
    return True

#%% Exercice 7
def occurrences(liste):
    d={}
    for x in liste:
        if x not in d:
            d[x]=1
        else:
            d[x]+=1
    return d

def cle_max(dico):
    valmaxi=-float('inf')
    for cle in dico:
        if dico[cle]>valmaxi:
            clemaxi=cle #clemaxi sera initialisée dès le 1er tour de boucle
            valmaxi=dico[cle]
    return clemaxi

def plus_frequent(phrase):
    liste=phrase.split(' ') #On crée une liste contenant tous les mots de la phrase
    d=occurrences(liste)
    return cle_max(d)

#%% Exercice 8
def anagrammes(mot1,mot2):
    d1=occurrences(mot1)
    d2=occurrences(mot2)
    return d1==d2

#%% Exercice 9
def longueurs(liste):
    d={}
    for mot in liste:
        n=len(mot)
        if n not in d:
            d[n]=[mot]
        else:
            d[n].append(mot)
    return d

#%% Exercice 10
def decalage(liste):
    n=len(liste)
    b=liste[0]
    for i in range(n-1):
        a=liste[i+1]
        liste[i+1]=b
        b=a
    liste[0]=b

#%% Exercice 11
def fusion(a,b):
    '''réalise la fusion de deux listes a et b déjà triées'''
    t=[]
    i=0;j=0

    while i<len(a) and j<len(b):  #la boucle tourne tant qu'au moins une des listes
    n'est pas parcourue entièrement
        if a[i]<b[j]:
            t.append(a[i])
            i=i+1
        else:
```

```
144                 t.append(b[j])
145                 j=j+1
146 
147         if i==len(a):       # s'il s'agit de la liste a qui est épuisée, on complète t
        avec les éléments restants de b
148             t=t+b[j:len(b)]
149         if j==len(b):       # s'il s'agit de la liste b qui est épuisée, on complète t
        avec les éléments restants de a
150             t=t+a[i:len(a)]
151 
152         return t
153 
```