Chapitre 1 : Structures de données avancées

Les Files ~ Corrigé 1.0

Exercice 1 :

```python
#Q1: créer fichier file.py

#Q2: creation d'une file vide
def creer_file():
    return []

#Q3
#nombre d'element dans f
def taille(f):
    return len(f)

#test si la file est vide
#Renvoie True si la file est vide et False sinon
def file_vide(f):
    return taille(f) == 0


#Q4: sommet de la file
#renvoie la tête de file (front en anglais):
#c'est le premier élément ajouté et non encore retiré
def sommet(f):
    assert not file_vide(f)
    return f[0]

#Q5: ajouter un element e à la fin de la file F (enqueue en anglais)
def enfiler(f, e):
    f.append(e)

#Q6: supprimer (retirer) le premier element de la file (dequeue en anglais)
def defiler(f):
    assert not file_vide(f)
    return f.pop(0)

#Q7: afficher tous les éléments de la file f.
def afficher(f):
    for i in range(taille(f)):
        x = defiler(f)
        print(x)
        enfiler(f,x)

#Q8: défiler la file f jusqu'à l'élément x
def defilerJusqua(f, x):
    while not file_vide(f) and sommet(f) != x :
        defiler(f)

```

```
46.   #Q9: Solution 1
47.   def appartient(f, e):
48.       """
49.       renvoie True si l'élément e appartient à la file f, False sinon.
50.       """
51.       result = False
52.       for i in range(taille(f)):
53.           if sommet(f) == e:
54.               result = True
55.           enfiler(f,defiler(f))
56.
57.       return result
58.
59.   #Q9: Solution 2
60.   def appartient(f, e):
61.
62.       f_temp = f.copy()
63.
64.       for i in range(taille(f_temp)):
65.           if sommet(f_temp) == e:
66.               return True
67.           defiler(f_temp)
68.
69.       return False
70.
71.   #Q10:
72.
73.   def inverser_file(f):
74.       """
75.       inverse les éléments de la file à laquelle elle est appliquée.
76.       """
77.       from pile5 import creer_pile,empiler,depiler,pile_vide
78.       p = creer_pile()
79.       while not file_vide(f):
80.           empiler(p,defiler(f))
81.       while not pile_vide(p):
82.           enfiler(f,depiler(p))
83.
84.
85.   #test
86.   if __name__ == '__main__':
87.       #Q2
88.       f = creer_file()
89.
90.       #Q3
91.       print("file vide :", file_vide(f))
92.
93.       #Q4
94.       if not file_vide(f):
95.           print("sommet de f:", sommet(f))
96.
97.       #Q5
98.       for i in range(10):
99.           enfiler(f,i)
100.
101.
```

```
#Q6
print("defiler f:", defiler(f))

#Q7
print("afficher f:" )
afficher(f)

#Q8
x = 6
defilerJusqua(f,6)
print("defilerJusqua "+str(x)+" :", f)

#Q9
e = 5
print(str(e)+ " appartient à f: ",appartient(f, e))

#Q10
print("file initiale:", f)
inverser_file(f)
print("file inversée:", f)
```

Exercice 2 :

```
#ex1 : File définie avec un tableau : file_ex2.py

#Q1: créer et renvoyer une file vide de taille n
def creerFile(n):
    return [0,0,[0]*n]
#2
def estVide(F):
    return F[0]-F[1]==0

#3
def estPleine(F):
    return F[0]-F[1]==len(F[2])

#4
def enfiler(F, x):
    assert not estPleine(F), "File Pleine"
    n=len(F[2])
    queue = F[0]
    F[2][queue % n]=x
    F[0]+=1

#5
def defiler(F):
    assert not estVide(F), "File vide"
    n=len(F[2])
    tete = F[1]
    indice = tete % n
    F[1]+=1
    return F[2][indice]
```

```python
#6
def sommet(F):
    assert not estVide(F), "File vide"
    n=len(F[2])
    tete = F[1]
    indice = tete % n
    return F[2][indice]


#test
if __name__ == '__main__':

    #Q1
    n = 5
    f = creerFile(n)
    print("f : ",f)

    #Q2
    print("f vide :",estVide(f))

    #Q3
    print("f pleine :",estPleine(f))

    #Q4
    import random
    try:
        for i in range(7):
            enfiler(f, random.randint(10,30))
            print("queue:",f[0], 'f',f)
            input()
    except:
        print("file pleine")
    print("f :",f)

    #Q5 & Q6
    print()
    print("defiler f:")
    defiler(f)
    print("f :",f)
    print("sommet de f:",sommet(f))

    print()
    print("defiler f:")
    defiler(f)
    print("f :",f)
    print("sommet de f:",sommet(f))

```

Exercice 3 :

```
1.  from file_ex2 import *
2.
3.  #Exercice 2: Nombres de Hamming
4.
5.  def Nombres_Hamming(n):
6.      f2 = creerFile(n)
7.      f3 = creerFile(n)
8.      f5 = creerFile(n)
9.      enfiler(f2, 1)
10.     enfiler(f3, 1)
11.     enfiler(f5, 1)
12.     for i in range (n):
13.         #On détermine le plus petit des trois têtes de file,
14.         #que l'on note k et que l'on imprime à l'écran ;
15.         k=min(sommet(f2),sommet(f3),sommet(f5))
16.         print(k)
17.         #On retire cet élément des files où il se trouve ;
18.         if sommet(f2)==k:
19.             defiler(f2)
20.         if sommet(f3)==k:
21.             defiler(f3)
22.         if sommet(f5)==k:
23.             defiler(f5)
24.         #on insère en queue des files f2, f3 et f5 les entiers 2k, 3k et 5k.
25.         enfiler(f2,2*k)
26.         enfiler(f3,3*k)
27.         enfiler(f5,5*k)
28.
29. #test
30. if __name__ == '__main__':
31.     n=18
32.     Nombres_Hamming(n)
```