2DS1 : Alternatives de correction

Exercice 1 ---------- 6pts

1) 2.5pts

```
from pile import *                    0.25pts   importation du module pile

def est_Dyck(ch,c1,c2):
    cmp=0                             0.5 pts   l'initialisation de/des compteurs
    for i in ch:
        if i== c1:                    0.5 pts   Incrémentation/décrémentation de/des compteurs
            cmp+=1
        elif i==c2:                   0.25pts   il faut que le mot commence par lettre 1
            if cmp>0:
               cmp-=1
            else:
                return False          0.5pts    lorsque lettre 2 excède lettre 1 dans le préfixe
        else:
            return False
    return cmp==0                     0.5 pts
```

2) 2pts

```
def est_DyckR(ch,c1,c2,c=0):  0.5pts      paramètres de la fonction
    if len(ch)==0:  return c==0  0.5 pts   conditions d'arrêt
    elif c<0 : return False
    else:
        if ch[0]==c1:
            c+=1
            return est_DyckR(ch[1:],c1,c2,c) 0.5pts
        elif ch[0]==c2 :
                c-=1
                return est_DyckR(ch[1:],c1,c2,c) 0.5 pts
        return False
```

Appels récursifs

3) 1.5pts

```
def est_DyckPile(ch,c1,c2):
    p=creerPile()          0.25pts
    for i in ch:           0.25 pts
```

```
        if i==c1:
            empiler(c1,p)  0.25pt
        elif i==c2:
            if not EstVide(p):
                depiler(p)                  0.25 pts
            else:
                return False   0.25 pts
        else:
            return False
    return EstVide(p) 0.25pts
```

Problème----------14pts

1)

```
def miseAjour(P,k,Q):                                   0.5pts
   P[2]=k*Q
```

2)

```
def afficheEtat(P):              1.5pts
    l=P[3]
    if l[0]==1:
        print(P[0]," est en attente")
    if l[1]==1:
        print(P[0]," est actif")
    if l[2]==1:
        print(P[0]," est interrompu")
```

3)2pts

```
def copy(F):
    FC=creerFile()   0.25
    ftmp=creerFile()  0.25
```

```
    while not EstVide(F):  0.25
        a=defiler(F)   0.25
        enfiler(a,FC)  0.25
        enfiler(a,ftmp)
    while not est_vide(ftmp):  0.25
        enfiler(defiler(ftmp),F) 0.25
    return FC 0.25
```

4) **1.5 pts**.

```
def Taille(f):
    if EstVide(f):          0.25pts
        return 0
    cmp=0                   0.25 pts
    fc=copy(f)              0.25
    while not EstVide(fc):  0.25
        defiler(fc)      0.25
        cmp+=1
    return cmp          0.25
```

5) 1.5 pts

```
def AppartientFile(F,ident):
    if EstVide(F):      0.25pts
        return False
    else:
      fc=copy(F)            0.25
      while not EstVide(fc):   0.25
        p=defiler(fc)               0.25
        if p[0]==ident:           0.25
            return True
```

```
    return False        0.25
```

6) 2pts

```
def Ajouter(F,q):
    while 1:                                    0.5
        n =input("id")
        if not AppartientFile(F,n): break
    while 1:
      try:
        te=int(input("te "))                    0.5
      except: continue
      else:
          if te>0: break
    tat=Taille(F)*q                             0.5
    t=(1,0,0)
    P=[n,te,tat,t]
    enfiler(P,F)                                0.5
```

7) 1.5 pts

```
def Reduire(F,Q):
    C=copy(F)                   0.5
    while not EstVide(F):       0.25
        defiler(F)
    while not EstVide(C):       0.25
        d=defiler(C)
        d[2]-=Q     0.25
        enfiler(d,F)  0.25
```

8) 2.5 pts

```
def Executer(F,Q):
```

```
    if EstVide(F):  0.25
        print("File Vide...")
    else:
      Reduire (F,Q)  0.5
      p=defiler(F) 0.25
      if p[1]-Q>0: 0.5
        p[1]-=Q     0.25
        p[3]=(0,0,1) 0.25
        enfiler(p,F)   0.5
```

9) 1 pts

```
def Affiche(F):
    c=copy(F)    0.5
    while not EstVide(c):
        p=depiler(c)     0.5
        afficheEtat(p)
```