Corrigé du TP Informatique 13

Exercice 1

1.
```
def est_libre(dispo_salle,deb) :
    for i in range(len(dispo_salle)) :
        if dispo_salle[i]<=deb :
            return i
    return -1
```

2.
```
def allocation(cours) :
    salles=[]
    dispo_salle=[]
    for k in range(len(cours)) :
        i=est_libre(dispo_salle,cours[k][0])
        if i>=0 :
            salles[i].append(cours[k])
            dispo_salle[i]=cours[k][1]
        else :
            salles.append([cours[k]])
            dispo_salle.append(cours[k][1])
    return salles
```

Exercice 2

1.
```
def fraction1(a,b) :
    x,res,k=a/b,[],1
    while x!=0 :
        while x<1/k :
            k+=1
        x=x-1/k
        res.append("+1/"+str(k))
    return res
```

   La fonction renvoie sans problème les deux premiers résultats demandés, mais pas le troisième : en effet, 2/5 n'est pas un nombre dyadique, sa représentation en mémoire conduit à une légère erreur, qui se répercute lors des itérations, et empêche le programme de terminer.

2. On esquive le problème en représentant x sous la forme d'un couple (p, q) où $x = \frac{p}{q}$. Les calculs sont font en nombres entiers en observant que $x - \frac{1}{k} = \frac{pk-q}{qk}$

```
def fraction2(a,b) :
    p,q,res,k=a,b,[],1
    while p!=0 :
        while p*k-q<0 :
            k+=1
        p,q=p*k-q,q*k
        res.append("+1/"+str(k))
    return res
```

En exécutant le programme avec les exemples précédents, les résultats s'obtiennent rapidement. Mais on rencontre une nouvelle difficulté : le dénominateur des fractions calculées grandit très rapidement, ce qui pose d'importants problèmes de complexité. Tester par exemple la fonction avec $x = \frac{5}{31}$.
Il existe des améliorations (non gloutonnes) de cet algorithme, et il est possible aussi de l'adapter pour déterminer le développement d'un rationnel $\geqslant 1$ en somme de fractions égyptiennes.