D.S. 1 I.P.T, solutions

1 Autour du calcul des décimales de e

a)
```
def factorielle(n):
    p=1
    for i in range(1,n+1):
        p=p*i
    return p
```

b) Pour `n=0` le `range(1,1)` est vide et on ne rentre pas dans la boucle `for`. Ainsi `p` reste à la valeur donnée par l'initialisation, i.e. `p=1`, et la fonction renvoie `1`.

c)
```
def SommeListe(L):
    S=0
    for i in range(len(L)):
        S=S+L[i]
    return S
```

d)
```
def ListInverseFact(n):
    L=[]
    for i in range(n+1):
        L.append(1/factorielle(i))
    return L
```

e)
```
def  approxiE1(n):
    return SommeListe(ListInverseFact(n))
```

f)
```
def approxiE2(n):
    S=0
    fact_courante=1
    for i in range(n+1):
        S=S+1/fact_courante
        fact_courante=fact_courante*(i+1)
    return S
```

g)
```
from fractions import *
def approxiE2Frac(n):
    S=0
    fact_courante=1
    for i in range(n+1):
        S=S+Fraction(1,fact_courante)
        fact_courante=fact_courante*(i+1)
    return S
```

h) Avec l'encadrement de e par u_n et $u_n + 1/(n.n!)$, on sait que l'écart entre u_{100} et e est inférieur à 10^{-159}. Donc les 159 premiers chiffres après la virgule dans u_{100} sont ceux de e.

Or la fraction `F` de l'énoncé nous donne une écriture exacte de u_{100} comme quotient de deux entiers. Pour avoir les 159 premiers chiffres après la virgule de u_{100}, il suffit donc de mutliplier le numérateur de u_{100} par 10^{159} et de faire la division euclidienne du résultat par le dénominateur de u_{100} ce qui donnera un entier à 160 chiffres qui seront les 160 premiers chiffres de l'écriture décimale de u_{100} donc de e (en comptant le 2 avant la virgule comme premier chiffre).

Ainsi en Python, on fait :

```
>>> F=approxiE2Frac(100)
>>> a=F.numerator
```

```
>>> b=F.denominator
>>> num=a*(10**159)
>>> Resultat= str(num//b) # l'avantage de le mettre sous forme str est de pouvoir vérifie
>>> Resultat
'271828182845904523536028747135266249775724709369995957496696762772407663035354759457138
>>>len(Resultat)
160
```

2 Rendu de monnaie

a)

```
def verifieDecroissant(L):
    for i in range(len(L)-1):
        if L[i]<L[i+1]:
            return False
    return True
```

b)
```
def MonMax(L):
    imax=0
    M=L[0]
    for i in range(len(L)):
        if L[i]>M:
            M=L[i]
            imax=i
    return M,imax
```

c)
```
def MaxEnTete(L):
    M,imax=MonMax(L)
    L[imax]=L[0]
    L[0]=M
```

d)
```
def rendre(valeur,liste):
    resultat=[]
    n=len(liste)
    monnaie=[]
    while valeur!=0:
        i=0
        while liste[i]>valeur:
            i=i+1
        monnaie.append(liste[i])
        valeur=valeur-liste[i]
    return monnaie
```

e) Avec l'algo. glouton, on obtient `monnaie= [50, 2, 2, 2, 2, 2, 2, 1]`
Or il y a une meilleure solution visible : `[20,20,20,2,1]`