



Alternative de Correction Concours Biologie et Géologie Epreuve d'Informatique

Barème sur 100

PROBLEME1 (55 pts)

Partie 1 :

1. 2.5 pts

$\pi_{nom}(\sigma_{idEtab = 'Libre' \text{ et } section = 'PC'}(Candidat))$

2. 2.5 pts

$\pi_{idC}(Candidat) - \pi_{idC}(Evaluation)$

Partie 2

3. 2.5 pts

Select **DISTINCTE**.nom from Etablissement E join candidat C on E.idEtab = C.idEtab
where C.section = 'BG' and E.idEtab <> 'Libre'

4. 2.5 pts

update Epreuve set duree= 2 where nomEpr='Informatique' and section= 'T';

5. 2.5 pts

select section, sum(coeff) from Epreuve group by section;

6. 3.75 pts

select C.nom, EP.nomEpr, EV.note from Candidat C, Epreuve EP, Evaluation EV where
C.idC=EV.idC and EP.idEpr=EV.idEpr and EP.date='2019/06/10' order by C.nom;

7. 3.75 pts

select idEpr, count(idC) from Evaluation where note >= 10 group by idEpr;

8. 5 pts

SELECT IdC FROM Evaluation WHERE note >= 15 GROUP BY IdC Having count(*) >= 3;
ou
select idC, count(idEpr) s from evaluation where note >= 15 group by idC having s >= 3;

9. 5 pts

select nom, dateNaiss from Candidat where sexe= 'F' ORDER BY dateNaiss DESC LIMIT
1;
ou

```

select nom, dateNaiss from Candidat where dateNaiss = (select max(dateNaiss) from
Candidat where sexe= 'F' )ANDsexe= 'F';
ou
select nom, dateNaiss from Candidat where dateNaiss= (select max(dateNaiss) from
(select nom, dateNaiss from Candidat where sexe= 'F'));

```

Partie 3 :

10.5 pts

```

def Score_candidat(cur, id):
    cur.execute('select sum(Ev.note*Ep.coef) from Epreuve Ep, Evaluation Ev
where Ep.idEpr=Ev.idepr and Ev.idC='+str(id))
    L=cur.fetchall()
    return L[0][0]

```

11.10 pts

```

def maj_candidat(cur):
    cur.execute('alter table Candidat add column score real')
    cur.execute('select idC from Candidat')
    L=cur.fetchall()
    for i in L:
        cur.execute('update Candidat set score = '+
str(score_candidat(cur,i[0])) + ' where idC =' + str(i[0]))

```

12.10 pts

```

def Admis(cur, d_nbplaces):
    d_admis={}
    for s in d_nbplaces:
        cur.execute('select idC from Candidat where section="'+s+'" order by score
desc;')
        L=cur.fetchall()
        L=[i[0] for i in L]
        d_admis[s]=L[:d_nbplaces[s]]
    return d_admis

```

PROBLEME 2 (45 points)

1. 2.5 pts

```

def IdRang(d_Rangs, s, rg):
    for i in d_Rangs[s]:
        if i[1]== rg:
            return i[0]

```

2. 3.75 pts

```

def listRangSection(d_Rangs, s):
    L=[] # ou bien:
    for i in d_Rangs:
        L.append(d_Rangs[i][1])
    L.sort()
    L=[e[1] for e in d_Rangs[s]]

```

```
return L
```

3. 2.5 pts

```
def MoyenneRang(L):  
    s=0  
    for i in range(len(L)):  
        s+=L[i]  
    return s/len(L)  
  
# ou bien:  
return sum(L)/len(L)
```

4. 3.75 pts

```
def MedianeRang(L):  
    n=len(L)  
    if n%2==1:  
        return L[n//2]  
    else:  
        return (L[n//2-1]+L[n//2])/2
```

5. 5 pts

Version 1

```
def DistributionRang(L):  
    Mo=MoyenneRang(L)  
    Me=MedianeRang(L)  
    if Mo==Me:  
        return 'symétrique'  
    elif Mo>Me:  
        return 'asymétrique à gauche'  
    else:  
        return 'asymétrique à droite'
```

Version 2 : avec retour d'une expression conditionnelle

```
def DistributionRang(L):  
    me=MedianeRang(L)  
    mo=MoyenneRang(L)  
    return 'symétrique' if me==mo else 'asymétrique à droite' if me > mo \  
    else 'asymétrique à gauche'
```

6. 10 pts

Version 1 :

```
def LQuartiles(L, q=1):  
    N=len(L)  
    if q==1:  
        r=(N-1)/4  
    elif q==2:  
        return MedianeRang(L)  
    else:  
        r=3*(N-1)/4  
    dr=r-int(r)  
    if dr==0:  
        return L[int(r)]  
    else:  
        a=L[int(r)]  
        b=L[int(r)+1]  
        if dr==0.25:  
            return (3*a+b)/4  
    elif dr==0.5:  
        return (a+b)/2  
    else:  
        #elif dr==0.75:  
        return (a+3*b)/4
```

Version 2 :

```
def LQuartiles(L, q=1):
```

```

    from math import ceil, floor
    N=len(L)
    r=q*(n-1)/4
    dr=r-int(r)
    if dr==0: #type(r)==int:
    Qq=L[int(r)]
    else:
        if dr==.75:
    Qq=(3*L[ceil(r)]+L[floor(r)])/4
    elif dr==.5:
    Qq=(L[ceil(r)]+L[floor(r)])/2
        else:
    Qq=(L[ceil(r)]+3*L[floor(r)])/4
    return Qq

```

7. 7.5 pts

Version 1 :

```

def ListAberrantes(L, k=3/2):
    LA=[]
    Q1=LQuartiles(L)
    Q3=LQuartiles(L, 3)
    E=Q3-Q1
    for e in L:
        if not (Q1-k*E <= e <= Q3+k*E):
    LA.append(e)
    return LA

```

Version 2 :

```

def ListAberantes(L, k=3/2):
    t=LQuartiles(L)
    Q1, Q3=t[0], t[2]
    LA=[]
    E=Q3-Q1
    ec1=Q1-k*E
    ec2=Q3+k*E
    i=0
    while L[i]<ec1:
    LA.append(L[i])
    i+=1
    i=len(L)-1
    while L[i]>ec2:
    LA.append(L[i])
    i-=1
    return LA

```

8. 10 pts

```

def Rapport(d_Rangs):
    f=open('rapport.txt', 'w')
    for s in d_Rangs:
        LR=listRangSection(d_Rangs, s)
        nb= len(LR)
        meillC= IdRang(d_Rangs, s, LR[0])
        Mo=MoyenneRang(LR)
        Me=MedianeRang(LR)
        dist=DistributionRang(LR)
        nbValAb=len(ListAberantes(LR))
        f.write(s+'*'+str(nb)+'*'+str(meillC)+'*'+str(Mo)+'*'+str(Me)+'*'+di
            st+'*'+str(nbValAbr)+'\n')
    f.close()

```