```python
# corTP10.py

import random as rd


## Q1
def etiquette_en_jeu(etiquette):
    a,b=etiquette.split('|')
    b=b.split(',')
    return a,[int(k) for k in b]

## Q2
def jeu_en_etiquette(jeu):
    texte=jeu[0]+'|'
    for k in jeu[1]:
        texte+=str(k)+','
    return texte[:-1]

## Q3
def terminaux(etiquette):
    joueur,jeu=etiquette_en_jeu(etiquette)
    if jeu[0:6]==[0,0,0,0,0,0] or jeu[7:13]==[0,0,0,0,0,0]:
        return True
    else:
        return False




## fonction affichage
def affichage(etiquette):
    joueur,jeu=etiquette_en_jeu(etiquette)
    print('   ' + '|' + str(jeu[-2]) + '|' + str(jeu[-3]) + '|' + str(jeu[-4]) + '|' + str(jeu[-5])
+ '|' + str(jeu[-6]) + '|' + str(jeu[-7]) + '|'+'   ')
    print(str(jeu[-1])+' '*(3-len(str(jeu[-1])))+'-'*13+' '*(3-len(str(jeu[6])))+str(jeu[6]))
```

```
033|     print('    ' + '|' + str(jeu[0]) + '|' + str(jeu[1]) + '|' + str(jeu[2]) + '|' + str(jeu[3]) +
'|' + str(jeu[4]) + '|' + str(jeu[5]) + '|'+'    ')
034|
035|
036|
037|
038|
039| ## Q4 ITERATIVE
040| def graphemancala_ite(profondeur=7,etiquette='0|4,4,4,4,4,4,0,4,4,4,4,4,4,0',S=[],A=[],E=[]):
041|     S=[0]
042|     prof_list=[0]
043|     A=[]
044|     E=[etiquette]
045|     list_attente=[0]
046|     indice=0
047|     while indice<len(list_attente):
048|         indice_sommet=list_attente[indice]
049|         prof=prof_list[indice_sommet]
050|         etiquette=E[indice_sommet]
051|         joueur,jeu=etiquette_en_jeu(etiquette)
052|         if (not terminaux(etiquette)) and prof<profondeur:
053|             if joueur=="0":
054|                 for s in range(6):
055|                     billes=jeu[s]
056|                     if billes!=0:
057|                         jeu_new=[k for k in jeu]
058|                         jeu_new[s]=0
059|                         for l in range(1,billes+1):
060|                             jeu_new[(s+l)%14]+=1
061|                         if (s+billes)%14!=6:
062|                             joueur_new="1"
063|                         else:
064|                             joueur_new="0"
065|                         if jeu_new[(s+billes)%14]==1 and 0<=(s+billes)%14<=5:
066|                             billes_recup=jeu_new[12-((s-billes)%14)]
```

```
067|                             jeu_new[12-((s-billes)%14)]=0
068|                             jeu_new[6]+=billes_recup
069|                         etiquette_new=jeu_en_etiquette((joueur_new,jeu_new))
070|                         ind_new_sommet=len(S)
071|                         S.append(ind_new_sommet)
072|                         A.append([indice_sommet,ind_new_sommet])
073|                         E.append(etiquette_new)
074|                         prof_list.append(prof+1)
075|                         list_attente.append(ind_new_sommet)
076|             if joueur=="1":
077|                 for s in range(7,13):
078|                     billes=jeu[s]
079|                     if billes!=0:
080|                         jeu_new=[k for k in jeu]
081|                         jeu_new[s]=0
082|                         for l in range(1,billes+1):
083|                             jeu_new[(s+l)%14]+=1
084|                         if (s+billes)%14!=13:
085|                             joueur_new="0"
086|                         else:
087|                             joueur_new="1"
088|                         if jeu_new[(s+billes)%14]==1 and 7<=(s+billes)%14<=12 :
089|                             billes_recup=jeu_new[12-(s+billes)%14]
090|                             jeu_new[12-(s+billes)%14]=0
091|                             jeu_new[13]+=billes_recup
092|                         etiquette_new=jeu_en_etiquette((joueur_new,jeu_new))
093|                         ind_new_sommet=len(S)
094|                         S.append(ind_new_sommet)
095|                         A.append([indice_sommet,ind_new_sommet])
096|                         E.append(etiquette_new)
097|                         prof_list.append(prof+1)
098|                         list_attente.append(ind_new_sommet)
099|
100|         indice=indice+1
101|     return S,A,E
```

```
102|
103|
104|
105| ## Q4 RECURSIVE
106| def ajout_sous_arbre(T,t,f):
107|     '''
108|     A REMPLIR
109|     '''
110|     S,A,E=T
111|     s,a,e=t
112|     l=len(S)
113|     for k in s:
114|         S.append(k+l)
115|     for (s1,s2) in a:
116|         A.append([s1+l,s2+l])
117|     E=E+e
118|     A.append([f,l])
119|     return S,A,E
120|
121|
122| def graphemancala_rec(n,etiquette='0|4,4,4,4,4,4,0,4,4,4,4,4,4,0',S=[],A=[],E=[],i=0):
123|     if i==0:
124|         S=[0]
125|         A=[]
126|         E=[etiquette]
127|     if n==i:
128|         return S,A,E
129|     joueur,jeu=etiquette_en_jeu(etiquette)
130|     p=len(S)-1
131|     if not terminaux(etiquette):
132|         if joueur=="0":
133|             for s in range(6):
134|                 billes=jeu[s]
135|                 if billes!=0:
136|                     jeu_new=[k for k in jeu]
```

```
                    jeu_new[s]=0
                    for l in range(1,billes+1):
                        jeu_new[(s+l)%14]+=1
                    if (s+billes)%14!=6:
                        joueur_new="1"
                    else:
                        joueur_new="0"
                    if jeu_new[(s+billes)%14]==1 and 0<=(s+billes)%14<=5:
                        billes_recup=jeu_new[12-((s-billes)%14)]
                        jeu_new[12-((s-billes)%14)]=0
                        jeu_new[6]+=billes_recup
                    etiquette_new=jeu_en_etiquette((joueur_new,jeu_new))
                    T=S,A,E
                    t=graphemancala_rec(n,etiquette_new,[0],[],[etiquette_new],i+1)
                    S,A,E=ajout_sous_arbre(T,t,p)
        if joueur=="1":
            for s in range(7,13):
                billes=jeu[s]
                if billes!=0:
                    jeu_new=[k for k in jeu]
                    jeu_new[s]=0
                    for l in range(1,billes+1):
                        jeu_new[(s+l)%14]+=1
                    if (s+billes)%14!=13:
                        joueur_new="0"
                    else:
                        joueur_new="1"
                    if jeu_new[(s+billes)%14]==1 and 7<=(s+billes)%14<=12 :
                        billes_recup=jeu_new[12-(s+billes)%14]
                        jeu_new[12-(s+billes)%14]=0
                        jeu_new[13]+=billes_recup
                    etiquette_new=jeu_en_etiquette((joueur_new,jeu_new))
                    T=S,A,E
                    t=graphemancala_rec(n,etiquette_new,[0],[],[etiquette_new],i+1)
                    S,A,E=ajout_sous_arbre(T,t,p)
```

```
172|     return S,A,E
173|
174|
175|
176| graphemancala=graphemancala_ite #je choisis quelle fonction j'utilise pour la suite
177|
178|
179|
180| ## Q5
181|
182| def sommets_controlés(S,A,E):
183|     cont=[]
184|     for k in range(len(E)):
185|         if E[k][0]=="0":
186|             cont.append(k)
187|     return cont
188|
189|
190|
191| ## Q6
192| def valeur(etiquette):
193|     joueur,jeu=etiquette_en_jeu(etiquette)
194|     if jeu[0:6]==[0,0,0,0,0,0] or jeu[7:13]==[0,0,0,0,0,0]:
195|         return jeu[6]+sum(jeu[0:6])-jeu[13]-sum(jeu[7:13])
196|     else:
197|         return jeu[6]-jeu[13]
198|
199|
200|
201|
202| ## Q7
203|
204| def marq_et_prop(x,value,compt,suc,pred,S0,F):
205|     """
206|     A REMPLIR
```

```
207|     """
208|     if x in S0 and x not in F:
209|         value[x]=max([value[s] for s in suc[x]])
210|     elif x not in S0 and x not in F :
211|         value[x]=min([value[s] for s in suc[x]])
212|     for u in pred[x]:
213|         compt[u]=compt[u]-1
214|         if compt[u]==0:
215|             value,compt=marq_et_prop(u,value,compt,suc,pred,S0,F)
216|     return value,compt
217|
218|
219|
220|
221| def predsuccesseur(S,A):
222|     """
223|     La fonction prend en paramètres :
224|     S: une liste d'entiers, qui correspond aux sommets d'un graphe
225|     A: une liste de tuples d'entiers (i,j), qui correspondent aux arêtes d'un graphe
226|
227|     La fonction renvoie un tuple contenant 4 listes:
228|     suc: une liste de listes, où chaque sous-liste contient les successeurs d'un sommet de S
229|     nbsuc: une liste contenant le nombre de successeurs pour chaque sommet de S
230|     pred: une liste de listes, où chaque sous-liste contient les prédécesseurs d'un sommet de S
231|     nbpred: une liste contenant le nombre de prédécesseurs pour chaque sommet de S.
232|     En utilisant les informations fournies par S et A, cette fonction permet de construire les
listes des successeurs et prédécesseurs pour chaque sommet du graphe, ainsi que le nombre de successeurs
et prédécesseurs pour chaque sommet.
233|     """
234|     suc=[[] for i in S]
235|     nbsuc=[0 for i in S]
236|     pred=[[] for i in S]
237|     nbpred=[0 for i in S]
238|     for (i,j) in A:
239|         suc[i].append(j)
```

```
            pred[j].append(i)
            nbpred[j]+=1
            nbsuc[i]+=1
        return suc,nbsuc,pred,nbpred
    
    
    
    
def minimax(S:list,A:list,E:list,S0:list,valeur)->list:
    '''
    A REMPLIR
    '''
    value=["" for i in S]
    suc,nbsuc,pred,nbpred=predsuccesseur(S,A)
    F=[]
    for k in range(len(nbsuc)):
        if nbsuc[k]==0:
            F.append(k)
    for i in range(len(F)):
        value[F[i]]=valeur(E[F[i]])
    compt=nbsuc
    for x in F:
        value,compt=marq_et_prop(x,value,compt,suc,pred,S0,F)
    L=[]
    for k in suc[0]:
        if value[k]==value[0]:
            L.append(k)
    return value, E[rd.choice(L)]
```

```
275|
276|
277| ## Q8
278|
279| def jouer(etiquette):
280|     billes=-1
281|     while billes<=0:
282|         if billes==0:
283|             print("mauvaise case, il n'y a plus de bille dans celle-ci ou c'est une case non
valide")
284|         s=input("quelle case jouée : ")
285|         s=int(s)
286|         if s>5:
287|             billes=0
288|         else:
289|             joueur,jeu=etiquette_en_jeu(etiquette)
290|             billes=jeu[s]
291|     jeu_new=[k for k in jeu]
292|     jeu_new[s]=0
293|     for l in range(1,billes+1):
294|         jeu_new[(s+l)%14]+=1
295|     if (s+billes)%14!=6:
296|         joueur_new="1"
297|     else:
298|         joueur_new="0"
299|     if jeu_new[(s+billes)%14]==1 and 0<=(s+billes)%14<=5:
300|         billes_recup=jeu_new[12-(s+billes)%14]
301|         jeu_new[12-(s+billes)%14]=0
302|         jeu_new[6]+=billes_recup
303|     etiquette_new=jeu_en_etiquette((joueur_new,jeu_new))
304|     return etiquette_new
305|
306|
307| ## Q9
308|
```

```
309| def game(etiquette='0|4,4,4,4,4,4,0,4,4,4,4,4,4,0'):
310|     affichage(etiquette)
311|     while not terminaux(etiquette):
312|         if etiquette[0]=='0':
313|             etiquette=jouer(etiquette)
314|             affichage(etiquette)
315|         else:
316|             print("Vador réfléchit")
317|             S,A,E=graphemancala(6,etiquette)
318|             v,etiquette=minimax(S,A,E,sommets_controlés(S,A,E),valeur)
319|             print("Vador joue:")
320|             affichage(etiquette)
321|             print("évaluation de l'état de la partie par le seigneur du mal : ",v[0])
322|     note=valeur(etiquette)
323|     if note >0:
324|         print("vous avez gagné!")
325|     elif note==0:
326|         print("match nul")
327|     else:
328|         print("vous avez perdu!")
329|
330|
331|
332|
```