Corrigés

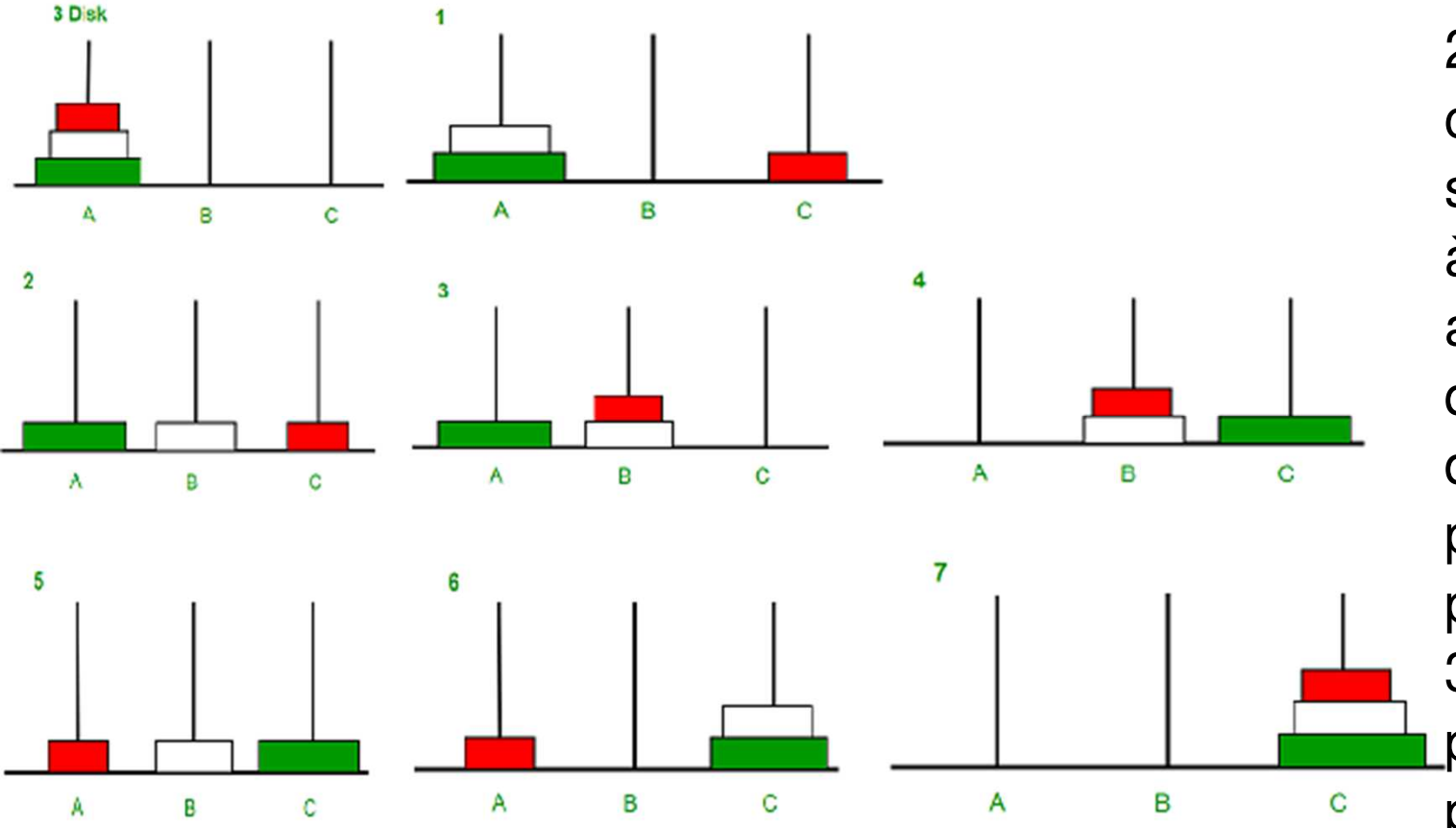
S4 – 3 Casse tête (Tour de Hanoï)

S6 – 3 Choix d'activité

TP S4 – 3. Casse tête (Tour de Hanoï)

Objectif du casse-tête :

Déplacer une pile de palets vers une autre pile, en obéissant aux règles simples suivantes



1) Un seul palet peut être déplacé à la fois.
2) Chaque déplacement consiste à prendre le palet supérieur d'une des piles et à le placer au-dessus d'une autre pile, c'est-à-dire qu'un palet ne peut être déplacé que s'il s'agit du palet le plus haut d'une pile.
3) Aucun palet ne peut être placé sur un palet plus petit.

TP S4 - 3. Casse tête (Tour de Hanoï)

casse_tete_0(A,B,C,3) déplacement pile(3) A vers C

casse_tete_0(A,C,B,2) déplacement pile(2) A vers B

casse_tete_0(A,B,C,1) → déplacement A vers C

casse_tete_0(A,C,B,1) → déplacement A vers B

casse_tete_0(C,A,B,1) → déplacement C vers B

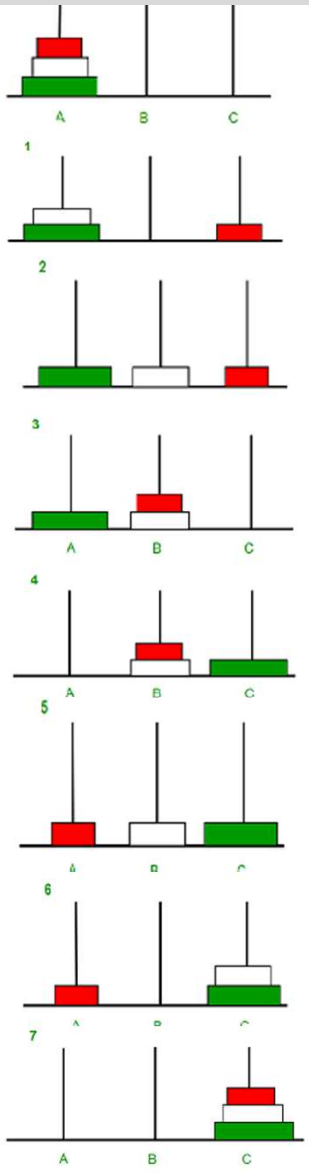
casse_tete_0(A,B,C,1) → déplacement A vers C

casse_tete_0(B,A,C,2) déplacement pile(2) B vers C

casse_tete_0(B,C,A,1) → déplacement B vers A

casse_tete_0(B,A,C,1) → déplacement B vers C

casse_tete_0(A,B,C,1) → déplacement A vers C




```
def casse_tete_0(pDep,pInt,pFin,n):
    # condition arret:
    if n == 1 :
        # deplacement
        pFin.append(pDep.pop())
        print(pDep,pInt,pFin)
    else:
        casse_tete_0(pDep,pFin,pInt,n-1)
        casse_tete_0(pDep,pInt,pFin,1)
        casse_tete_0(pInt,pDep,pFin,n-1)
```

[3, 2]	[ ]	[1]
[3]	[1]	[2]
[ ]	[3]	[2, 1]
[ ]	[2, 1]	[3]
[2]	[3]	[1]
[ ]	[1]	[3, 2]
[ ]	[ ]	[3, 2, 1]

TP S6 - 3.UN CHOIX D'ACTIVITÉ

- *Le choix glouton consiste à choisir la conférence qui termine le plus tôt possible, préservant ainsi la disponibilité de la salle pour les conférences ultérieures.*

- Conférences référencées sous forme de tuples formés comme suit:

(NomConf, début, fin)

```
liste_confJ1 = [ ('C1',8,9) , ('C2',8.75,10.25) , ('C3',8.25,11.5) ,
('C4',11.25,12.25) , ('C5',13,14.5) , ('C6',11,15) , ('C7',13.25,15.5) , ('C8',17,18),
('C9',15.75,18.5) ,('C10',17.25,18.75)  ]
```

- ***Ecrire*** *le script d'une fonction* `choix_conferences(listeC: list)->list` *qui renvoie une liste de conférences en maximisant le nombre de celles-ci à partir d'une liste de tuples de conférences possibles,* `listeC`.

- ***Tester*** *votre fonction pour la journée 1 du colloque.*

3. ILLUSTRATION CLASSIQUE: UN CHOIX D'ACTIVITÉ

```
liste_confJ1 = [('C1',8,9), ('C2',8.75,10.25) , ('C3',8.25,11.5), ('C4',11.25,12.25),
('C5',13,14.5), ('C6',11,15), ('C7',13.25,15.5), ('C8',17,18), ('C9',15.75,18.5),
('C10',17.25,18.75)]

def choix_conferences_it(listeC:list)->list :
    ''' maximiser le nombre de conférences dans listeC
        précondition: conférences triées par heure de fin croissante '''
    journée = [listeC[0]]
    fin = listeC[0][2]
    for i in range(1,len(listeC)):
    if listeC[i][1] > fin: # début conf i > fin(journée)
        journée.append(listeC[i])
        fin = listeC[i][2]
    return journée
```

3. ILLUSTRATION CLASSIQUE: UN CHOIX D'ACTIVITÉ

```
def choix_conferences_rec(listeC:list)->list :
    # condition arrêt
    if listeC[-1][1] < listeC[0][2]: # début dernière conf < fin  1ère conf
        return [listeC[0]]
    i = 1
    while listeC[i][1] < listeC[0][2]:
       i += 1
    return [listeC[0]] + choix_conferences_rec(listeC[i:])
```

Schéma d'appel pour

```
liste_confJ1 = [('C1',8,9), ('C2',8.75,10.25) , ('C3',8.25,11.5),
('C4',11.25,12.25), ('C5',13,14.5), ('C6',11,15)] ?
```

3. ILLUSTRATION CLASSIQUE: UN CHOIX D'ACTIVITÉ

Schéma d'appel pour

```
liste_confJ1 = [('C1',8,9), ('C2',8.75,10.25) , ('C3',8.25,11.5),
('C4',11.25,12.25), ('C5',13,14.5), ('C6',11,15)]
```