Métropole - septembre 2024 - sujet 2 (corrigé)

Exercice 1 (Programmation Python et décidabilité)
Partie A : boucle `while`

1. : La variable `i` prend successivement les valeurs `7`, `8`, `9` et `10`, donc `f1(7)` se termine
2. `f1(-2)` se termine et la valeur renvoyée est `10`
3. Le cinq premières valeurs prises par la variable `i` sont `12` ,`13`, `14`, `15` et `16`. Ainsi, `f1(12)` ne se termine pas (car toujours `i` prend toujours une valeur différent de `10`)
4. D'après ce qui précède, `f1(n)` se termine pour tous les entiers strictement inférieurs à `11`.

Partie B : fonction récursive

5. `f2(4)` se termine et renvoie `6`
6. `f2(5)` ne se termine pas, car nous avons les appels récursifs suivants : `f2(3)`, `f2(1)`, `f2(-1)`, `f2(-3)`, etc. En particulier, on ne rencontre jamais le cas de base `n=0`
7. L'appel `f2(n)` se termine pour tous les entiers pairs supérieurs ou égaux à `0`.
8. La fonction suivante convient :

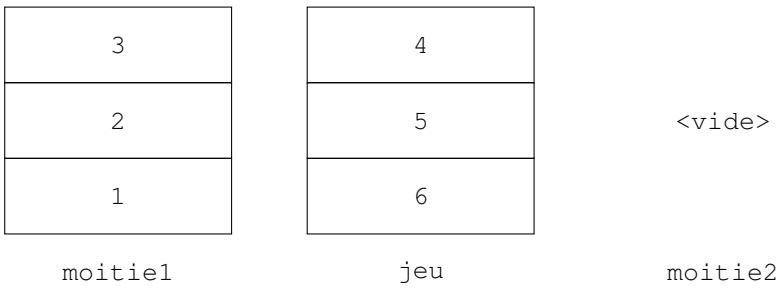
```
def infini(n):
    return n + infini(n-1)
```

Partie C : le problème de l'arrêt

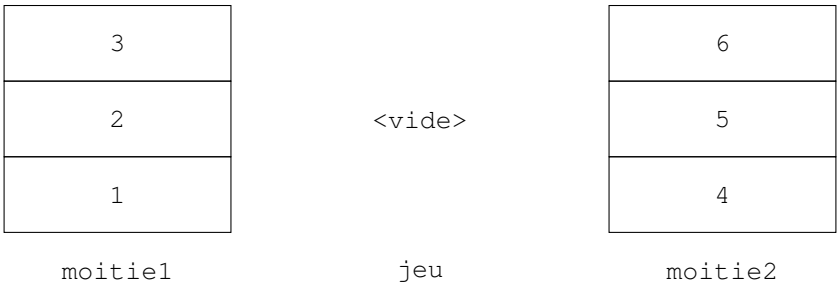
9. On appelle `infini(42)` et donc `paradoxe(code_paradoxe)` ne termine pas.
10. On exécute `return 0`, donc `paradoxe(code_paradoxe)` termine.
11. Il y a un paradoxe (la fonction `arret` se termine quand elle ne termine pas), donc la fonction `arret` ne peut pas exister.

Exercice 2 (Programmation Python, POO et pile)

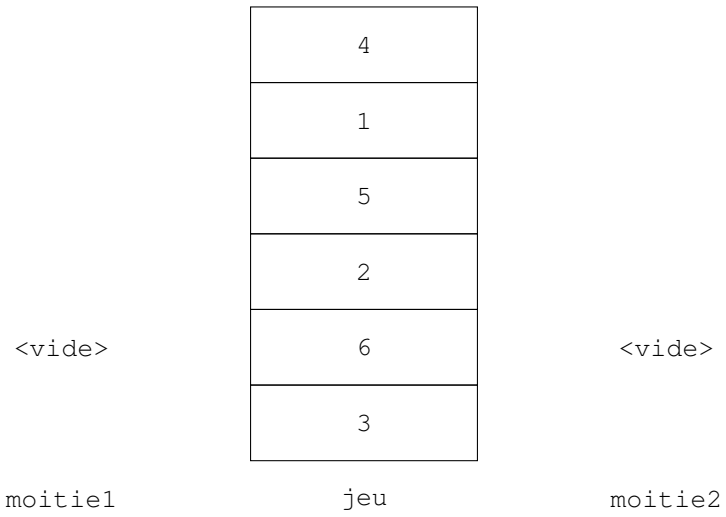
1. On a les trois étapes suivantes :
   - ⋆ Etape 1 :



   - ⋆ Etape 2 :



   - ⋆ Etape 3 :



2. Le code suivant convient :

```
def produire_jeu(n):
    resultat = Pile()
    for i in range(n):
        resultat.empile(n - i)
    return resultat
```

3. On a les erreurs suivantes :
   - ⋆ ligne 3 : il faut écrire `Pile()`
   - ⋆ ligne 4 : il faut écrire `for i in range(n//2)`
   - ⋆ ligne 6 : il faut écrire `for i in range(n//2)`

4. La fonction suivante convient :

```
def recombiner(m1, m2):
    m = Pile()
    while not m1.est_vide():
        m.empile(m1.depile())
        m.empile(m2.depile())
    return m
```

5. La fonction suivante convient :

```
def faro(p, n):
    p1, p2 = scinder_jeu(p, n)
    return recombiner(p1, p2)
```

6. Le jeu de tests suivant convient :

```
p1 = Pile()
p1.empile(1)
p2 = Pile()
assert not identiques(p1, p2)
p3 = Pile()
p3.empile(5)
p4 = Pile()
p4.empile(8)
assert not identiques(p3, p4)
p5 = Pile()
p5.empile(5)
p6 = Pile()
p6.empile(5)
assert identiques(p5, p6)
```

7. La fonction suivante convient :

```
def ordre_faro(n):
    compteur = 1
    p_init = produire_jeu(n)
    m = faro(produire_jeu(n), n)
    while not identiques(p_init,m):
        compteur = compteur + 1
        m = faro(m,n)
    return compteur
```

Exercice 3 (Réseaux, protocoles de routage et bases de données)

Partie A : configuration réseau dans la DMZ

1. Une adresse IPv4 est constituée de quatre octets.
2. ⋆ Le `Serveur_web` a pour adresse `172.16.0.1`
   ⋆ Le `Serveur_BDD` a pour adresse `172.16.0.2`
3. La commande `ping` permet de tester la connexion entre deux machines.
4. La passerelle par défaut est mal configurée, il faut indiquer l'adresse IP `192.168.1.254` et non `192.168.0.254` qui ne correspond pas au réseau local de `PC_A1`.

Partie B : routage

5. On a le chemin suivant : `PC_A1` → Routeur `A` → Routeur `B` → Routeur `C` → Routeur D → `Serveur_impression`
6. On a le chemin suivant : `PC_A1` → Routeur `A` → Routeur `B` → Routeur `C` → ? ?. Les paquets n'arrivent donc pas à destination.
7. On a la table de routage suivante :

Routeur `C`		
Destination	Prochain saut	Métrique
`172.16.0.0`	`10.0.2.2`	2
`192.168.0.0`	`10.0.2.2`	2
`192.168.1.0`	`10.0.2.2`	2
`192.168.2.0`	`10.0.3.2`	1
`192.168.3.0`	`10.0.4.2`	1
`10.0.0.0`	`10.0.2.2`	1
`10.0.1.0`	`10.0.2.2`	1
`10.0.2.0`	–	–
`10.0.3.0`	–	–
`10.0.4.0`	–	–
`10.0.5.0`	`10.0.3.2`	1
`0.0.0.0`	`10.0.2.2`	2

8. On a le chemin suivant : `PC_A1` → Routeur `A` → Routeur `B` → Routeur `C` → Routeur `D` → `Serveur_impression`
9. Le débit de la liaison entre les routeurs `C` et `D` étant faible (10 Mb/s), il vaut mieux traverser un routeur de plus en passant par le routeur `E` car les débits sont meilleurs (1 Gb/s).
10. Il faut faire les deux modifications suivantes :

Destination	Prochain saut	Métrique
`192.168.2.0`	`10.0.4.2`	2
`10.0.5.0`	`10.0.4.2`	1

On a le chemin suivant : `PC_A1` → Routeur `A` → Routeur `B` → Routeur `C` → Routeur `E` → Routeur `D` → `Serveur_impression`

Partie C : exploitation de la base de données

11. La requête suivante convient :

```
SELECT titre_parution FROM parution
```

12. Cette requête permet de sélectionner le numéro de parution et le numéro des pages qui ont pour police de caractère `Arial,12`. Ces informations seront classées par ordre croissant de numéro de parution
13. La requête suivante convient :

```
SELECT num_image, titre_image, poids
FROM image
WHERE poids > 1000
```

14. Cette requête permet d'obtenir les numéros des parutions qui contiennent des images avec le terme `Appolo` dans leur titre.
15. Cette requête permet d'ajoute une image (numéro 2923) à la table `image`. Cette image a pour titre `Volcans du massif central`, une largeur et une hauteur de 400 pixels et un poids de 1430 Ko.
16. La requête suivante convient :

```
INSERT INTO texte VALUES(2754, 'Vulcania', 'Parc dattraction', 250)
```

17. Cette requête permet de supprimer de la table `texte` le texte ayant pour numéro 2034
18. La requête suivante convient :

```
DELETE FROM comporte_texte WHERE num_texte = 2034
```