Mayotte - mai 2022 - sujet 2 (corrigé)

Exercice 1 (Piles)

1. On a le schéma suivant :

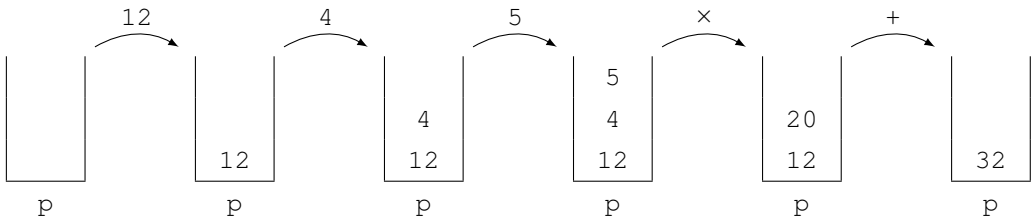
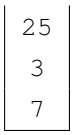


Schéma descriptif des différentes étapes d'exécution

2. (a) La fonction `top` renvoie la valeur au sommet de la pile. Donc, La variable `temp` contient la valeur `25`.

   (b) On a la pile suivante :



3. La fonction suivante convient :

```
def addition(p):
    v1 = depiler(p)
    v2 = depiler(p)
    empiler(p, v1+v2)
```

4. Le code suivant convient :

```
p = pile_vide()
empiler(p,3)
empiler(p,5)
addition(p)
empiler(p,7)
multiplication(p)
```

Exercice 2 (Bases de données et SQL)

1. Le couple `(NumClient, NumChambre)` ne peut pas jouer le rôle de clé primaire, car un client peut réserver plusieurs fois la même chambre (à des dates différentes).
2. (a) La requête suivante convient :

```
SELECT Nom, Prenon FROM Clients
```

   (b) La requête suivante convient :

```
SELECT Telephone FROM Clients WHERE Nom = 'Hopper' AND Prenom = 'Grace'
```

3. La requête suivante convient :

```
SELECT NumChambre
FROM Reservations
WHERE date(DateArr) <= date('2020-12-28') AND date(DateDep) > date('2020-12-28')
```

4. (a) La requête suivante convient :

```
UPDATE Chambres
SET Prix = 75
WHERE NumChambre = 404
```

   (b) La requête suivante convient :

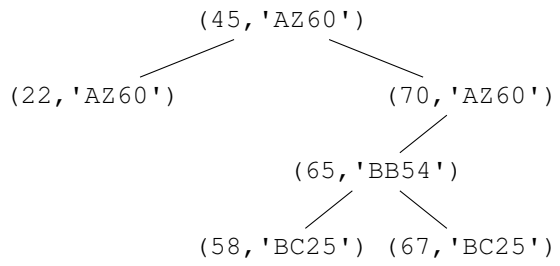
```
SELECT Reservations.NumChambre
FROM Reservations
JOIN Clients ON Reservations.NumClient = Clients.NumClient
WHERE Prenom = 'Edgar' AND Nom = 'Codd'
```

Exercice 3 (Binaire et systèmes d'exploitation)

1. (a) Un octet correspond à 8 bits.
   (b) Il est possible de coder $2^8 = 256$ valeurs.
   (c) Ces 256 valeurs sont comprises entre 0 et 255.
2. (a) Comme $65 = 2^6 + 2^0$, on a $65 = (0100\ 0001)_2$.
   (b) On a directement $0011\ 1010 = 2^5 + 2^4 + 2^3 + 2^1 = 32 + 16 + 8 + 2 = 58$.
   (c) L'écriture en base 2 de -58 est donnée par $1100\ 0101 + 0000\ 0001$, soit $1100\ 0110$.
   (d) La valeur binaire de $65 - 58$ est donc $0100\ 0001 + 1100\ 0110 = 0000\ 0111$ (avec 1 en bit de retenu.
3. (a) Il faut saisir l'instruction : `mv ./pierre/documents/saxo.mp3 ./pierre/musiques/saxo.mp3`
   (b) Il faut saisir l'instruction : `mv ./pierre/bizarre ./pierre/videos`

Exercice 4 (Arbres binaires de recherche)

1. (a) On a l'arbre suivant :



   (b) Pour obtenir la liste triée, il faut effectuer un parcours en profondeur infixe.

2. La fonction suivante convient :

```
fonction taille(a)
    si a est null alors
        renvoyer 0
    sinon
        renvoyer 1 + filsgauche(a) + filsdroit(a)
    fin si
fin fonction
```

3. (a) Cette fonction permet de rechercher un billet de numéro `n` dans un arbre `a`. Si le billet recherché est présent dans l'arbre, la fonction renvoie `Vrai` ; dans le cas contraire, la fonction renvoie `Faux`.

   (b) La fonction suivante convient :

```
fonction mystereABR(a, n)
    si a est null alors
        renvoyer Faux
    sinon si billet(a) vaut n alors
        renvoyer Vrai
    sinon si billet(a) strictement inférieur à n alors
        renvoyer mystereABR(filsgauche(a))
    sinon
        renvoyer mystereABR(filsdroit(a))
    fin si
fin fonction
```

Exercice 5 (Algorithme et programmation générale)

1. La fonction suivante convient :

```
def autre(x):
    if x == 0:
        return 1
    if x == 1:
        return 0
```

2. (a) La fonction suivante convient :

```
def nbValeurs(li, v):
    nb_colonne = 10
    cpt = 0
    for i in range(nb_colonne):
        if grille[li][i] == v:
            cpt = cpt + 1
    return cpt
```

(b) La fonction suivante convient :

```
def regle1(li):
    nb_colonne = 10
    v = -1
    if nbValeurs(li, 0) == 5:
        v = 1
    if nbValeurs(li, 1) == 5:
        v = 0
    if v != -1:
        for i in range(nb_colonne):
            if grille[li][i]== -1:
                grille[li][i] = v
```

3. La fonction suivante convient :

```
def regle3(li):
    for col in range(8):
        if grille[li][col] == grille[li][col+2] and grille[li][col+1] == -1:
            grille[li][col+1] = autre(grille[li][col])
```

4. La fonction suivante convient :

```
def convert(L):
    numbit = len(L)-1
    s = 0
    for n in L:
        s = s + n*2**(numbit)
        numbit = numbit - 1
    return s
```

5. La fonction suivante convient :

```
fonction doublon(L) :
  pour tous les éléments v1 de L
    compteur ← 0
    pour tous les éléments v2 de L
      si v1 = v2
        compteur ← compteur + 1
      fin si
    fin pour
    si compteur > 1
      renvoyer Vrai
    fin si
  fin pour
  renvoyer Faux
fin fonction
```