

Éléments de correction du bac blanc

Exercice 1 :

1

```
def __init__(self, nature, surface, prix_moy):
    self.nt = nature
    self.sf = surface
    self.pm = prix_moy
```

2

L'instruction `b1.estim_prix()` renvoie 140000.0 (estimation du prix de `b1`) ; la valeur renvoyée est de type flottant, car `self.sf` et `self.pm` sont de type flottant.

3

```
def estim_prix(self):
    prix_brut = self.sf * self.pm
    if self.nt=='maison':
        return prix_brut*1.1
    elif self.nt=='bureau':
        return prix_brut*0.8
    else :
        return prix_brut
```

4

```
def nb_maison(lst):
    compteur = 0
    for b in lst :
        if b.nt=='maison':
            compteur = compteur + 1
    return compteur
```

5a

on effectue un parcours infixe : `b2 - b4 - b1 - b5 - b3 - b6`

5b

```
def contient(surface,abr):
    if abr.est_vide():
        return False
    elif abr.get_v().sf >= surface:
        return True
    else :
        return contient(surface, abr.get_d())
```

Exercice 2 :

1

R2 (utilisation de *)

2a

```
SELECT nom, avis
FROM Client
INNER JOIN Reservation ON Client.idClient = Reservation.idClient
WHERE jour = '2021-06-05' AND heure = '19:30:00'
```

2b

```
SELECT nom
FROM Plat
INNER JOIN Commande ON Plat.idPlat = Commande.idPlat
INNER JOIN Reservation ON Reservation.idReservation =
Commande.idReservation
WHERE (Categorie = 'plat principal' OR Categorie = 'dessert') AND jour =
'2021-04-12'
```

3

Cette requête permet d'ajouter un plat qui aura pour idPlat 58, pour nom "Pêche Melba", pour catégorie "dessert", pour description "Pêches et glace vanille" et pour prix 6,5 euros.

4a

```
DELETE FROM Commande
WHERE idReservation = 2047
```

4b

```
UPDATE Plat
SET prix = prix+0.05*prix
WHERE prix < 20.0
```

Exercice 3 :

Partie A

1

```
lab2[1][0] = 2
```

2

```
def est_valide(i,j,n,m):  
    return i>=0 and j>=0 and i<n and j<m
```

3

```
def depart(lab):  
    n = len(lab)  
    m = len(lab[0])  
    for i in range(n):  
        for j in range(m):  
            if lab[i][j]==2:  
                return (i,j)
```

4

```
def nb_cases_vides(lab):  
    n = len(lab)  
    m = len(lab[0])  
    compt = 0  
    for i in range(n):  
        for j in range(m):  
            if lab[i][j]==2 or lab[i][j]==3 or lab[i][j]==0:  
                compt = compt + 1  
    return compt
```

Partie B

1

L'appel de la fonction renvoie : [(2, 2), (1, 1)]

2a

```
# entrée: (1, 0), sortie (1, 5)
chemin = [(1, 0)]
chemin.append((1,1))
chemin.append((2,1))
chemin.pop()
chemin.append((1,2))
chemin.append((1,3))
chemin.append((2,3))
chemin.append((3,3))
chemin.append((3,4))
chemin.pop()
chemin.pop()
chemin.pop()
chemin.append((1,4))
chemin.append((1,5))
```

```
def solution(lab):
    chemin = [depart(lab)]
    case = chemin[0]
    i = case[0]
    j = case[1]
    while lab[i][j] != 3:
        lab[i][j]=4
        v = voisines(i,j,lab)
        if len(v) != 0 :
            prochaine = v.pop()
            chemin.append(prochaine)
            i = prochaine[0]
            j = prochaine[1]
        else :
            chemin.pop()
            n = len(chemin)
            i = chemin[n-1][0]
            j = chemin[n-1][1]
    return chemin
```