Exercice 1 : Programmation orientée objet

1) Le code de la classe **Bim** complété est :

```
class Bim :
    def __init__(self,nature,surface,prix_moy) :
      self.nt = str(nature)
      self.sf=float(surface)
      self.pm=float(prix_moy)
    def estim_prix(self) :
      return self.sf*self.pm
```

2) L'instruction

```
b1=Bim('maison',70.0,2000.0)
b1.estim_prix()
```

Renvoi une valeur de type **float()**.

3) L'affination de la classe **Bim** est :

```
class Bim :
    def __init__(self,nature,surface,prix_moy) :
      self.nt = str(nature)
      self.sf=float(surface)
      self.pm=float(prix_moy)
    def estim_prix(self) :
       if self.nt== "maison" :
            return self.sf*self.pm*1.1
       elif self.nt=="bureau" :
            return self.sf*self.pm*0.8
       else :
             return self.sf*self.pm
```

4) La fonction **nb_maison(lst)** prenant en argument une liste de biens immobiliers et qui renvoie le nombre d'objets de nature **"maison"** contenu dans la liste est :

```
class Bim :
    def __init__(self,nature,surface,prix_moy) :
      self.nt = str(nature)
      self.sf=float(surface)
      self.pm=float(prix_moy)
    def estim_prix(self) :
       if self.nt== "maison" :
            return self.sf*self.pm*1.1
       elif self.nt=="bureau" :
            return self.sf*self.pm*0.8
       else :
             return self.sf*self.pm
#_____________________________________________________________________
    def nb_maison(lst) :
          nb_maison_dans_lst=0
          for i in range(len(lst)) :
              if lst[i]== "maison" :
                    nb_maison_dans_lst+=1
          return nb_maison_dans_lst
```

5)a)

La lise des biens triés dans l'ordre croissant de leur surface est : b4<b2<b1<b5<b3<b6.

5)b)

```
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: SQL

1) La requête qui correspond au besoins est la requête R2.

2) a)

```
SELECT * FROM RESERVATION WHERE jour="2021-06-05" AND heure="19:30:00";
```

2) b)

```
SELECT * FROM Plat, Reservation WHERE (categorie="plat principal" AND jour="2021-04-12") OR (categorie="dessert" AND jour="2021-04-12");
```

3)

Cette variable insère un nouveau plat dans le tableau Plat avec les informations suivantes : (identifiant <idPlat>=58, nom <nom>="Pêche Melba", catégorie <categorie>="dessert", description <description>="pêches et glace vanille", prix <prix>=6.5)

4) a)

```
DROP Plat WHERE idReservation=2047;
```

4) b)

```
UPDATE prix FROM Plat (prix*5/100)+prix WHERE prix<20.00;
```

Exercice 3: Les réseaux

1) a) Les adresses des réseaux L1 et L2 sont : R1 pour L1 avec 10.1.1.0/24 comme adresse externe et R6 pour L2 avec 10.1.7.2/24
   b) La plus petite et la plus grande adresse IP pour :

R1 est :	
La plus petite	La plus grande
10.1.1.2/24	192.168.1.1/24

R6 est :	
La plus petite	La plus grande
10.1.7.2/24	172.16.0.1/16

   c) On peut connecter 65536-2[pour les deux ports administateurs] =65534 ordinateurs sur le réseau local L1, le chiffre est le même pour le réseau local L2.
2) a) L'avantage d'avoir plusieurs chemins possible pour relier les deux réseaux est que si un routeur ne fonctionne plus (par exemple, le routeur 3) il y a d'autres chemins permettant d'accéder au réseau local 2. Par exemple si le chemin L1→R2→R3→R4→R5→L2 ne fonctionne pas, le chemin L1→R2→R5→L2 reste encore accessible.
   b) Le chemin le plus court pour relier le réseau L1 et L2 est R1→R2→R5→R6, soit quatre saut en comptant le routeur de départ et le routeur d'arrivé.
   c)

Ether	10^7
FastEther	10^8
coût	10^8/Ether or FastEther

Le chemin reliant R1 et R6 ayant le plus petit coût est R1→R2→R3→R4→R5→R6 soit $\frac{10^8}{10^7}+\frac{10^8}{10^8}+\frac{10^8}{10^8}+\frac{10^8}{10^8}+\frac{10^8}{10^7}=23$

3)

IP réseau	Passerelle suivante	Interface
10.1.3.0/24	10.1.3.2	Interface 1
10.1.4.0/24	10.1.4.2	Interface 2
10.1.6.0/24	10.1.6.2	Interface 3
10.1.7.0/24	10.1.7.1	Interface 4
10.1.9.0/24	10.1.9.2	Interface 5
10.1.10.0/24	10.1.10.2	Interface 6

IP réseau de destination	Passerelle suivante	Interface
172.16.0.0/16	172.16.0.1	Interface 1
172.17.0.0	172.17.0.1	Interface 2
172.19.0.0	172.19.0.1	Interface 3

Exercice 4: Les systèmes d'exploitation

Partie A :

1) Sachant que les processus 1, 2 et 3 s'exécutent de manière concurrente, comme il n'y a que trois processus évoqués dans le code des programmes (table traçante, modem et imprimante) sachant que chaque programme demande l'un de ces trois processus en étape 1, Programme 1 demande table traçante, Programme 2 demande modem et Programme 3 demande imprimante. A l'étape 2, sachant que les trois processus disponibles sont pris respectivement par les trois programmes, aucun ne peut avancer car le second processus dont il a besoin est déjà pris par l'un de ses voisins.

2) La modification de l'ordre d'exécution pour le programme 3 pour éviter l'interblocage est :

Programme 3	
Avant modification	Après modification
demander (imprimante) demander (table traçante) exécution libérer (table traçante) libérer (imprimante)	demander (table traçante) demander (imprimante) exécution libére (imprimante) libérer (table traçante)

3) Sachant que le programe 3 détient déjà la table traçante, comme deux programmes ne peuvent avoir un même processus de manière simultanée alors l'état du programme 1 sera dans l'état bloqué, soit la réponse b).

Partie B :

1) Parmi les quatre commandes suivantes, celle permettant l'affichage est ps -ef soit la réponse b).
2) L'identifiant du processus parent à l'origine de tous les processus concernant le navigateur web (chromium-browser) est 831.
3) L'identifiant du processus dont le temps est le plus long est 6211 avec un temps 01 :16.

Exercice 5: structures de données linéaires

1) c) Parmi les propositions suivantes, la structure de données la plus appropriée pour la mise en œuvre de FIFO est la pile, soit la réponse c)

2) La fonction permettant d'ajouter un nouveau processus en attente est:

```
def ajouter(lst,proc):
    lst.append(proc)
    return lst
```

3)

Le résultat de l'exécution de la séquence d'instructions suivante est :

4)

a) La fonction est_vide(f) de type booléen est :

```
def est_vide(f) :
    if len(f)==0 :
        return True
    else :
        resturn False
```

b) La fonction enfiler(f,elt) est :

```
def enfiler(f,elt) :
     f.append(elt)
     return f
```

c) La fonction defiler(f) renvoyant l'élément défilé est :

```
def defiler(f) :
    print(f.pop(0))
    return f
```