Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

Exercice 1 - Base de données du parc informatique d'un lycée

1.a.

salle	marque_ordi
012	HP
114	Lenovo
223	Dell
223	Dell
223	Dell

1.b.

nom_ordi	salle
Gen-24	012
Tech -62	114
Gen-132	223

2.

```
SELECT * FROM Ordinateur WHERE annee >= 2017 ORDER BY annee ASC ;
```

3.a. Des ordinateurs différents peuvent avoir le même numéro de salle or une clé primaire doit permettre d'identifier de façon unique chaque enregistrement de la table.

3.b. Schéma relationnel de la relation `Imprimante` :

`Imprimante(nom_imprimante, marque_imp, modele_imp, salle, #nom_ordi)`

4.a.

```
INSERT INTO Videoprojecteur VALUES (315, 'NEC', 'ME402X', false) ;
```

4.b.

```
SELECT Ordinateur.salle, Ordinateur.nom_ordi, Videoprojecteur.marque_video
FROM Ordinateur JOIN Videoprojecteur ON Ordinateur.salle = Videoprojecteur.salle
WHERE Ordinateur.video = true AND Videoprojecteur.tni = true ;
```

Ou avec les alias :

```
SELECT O.salle, O.nom_ordi, V.marque_video
FROM Ordinateur AS O JOIN Videoprojecteur AS V ON O.salle = V.salle
WHERE O.video = true AND V.tni = true;
```

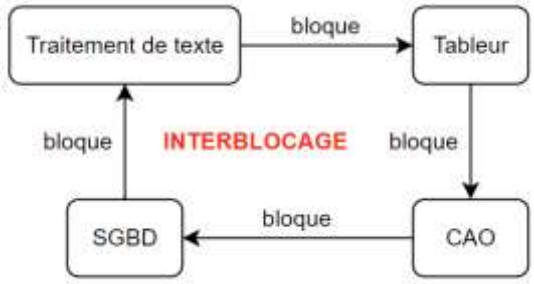
Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

Exercice 2 - SoC, Gestion des processus et Routage

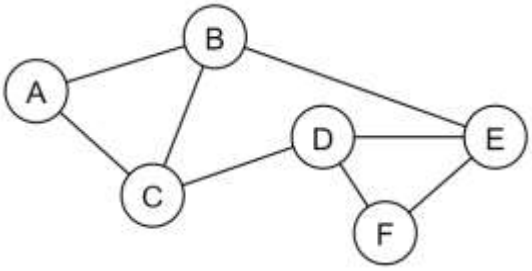
1. Les systèmes sur puces permettent une plus grande intégration grâce à leur faible encombrement. La proximité des composants permet une plus grande rapidité de traitement et une consommation énergétique réduite. Enfin l'automatisation de leur production permet d'obtenir des coûts réduits.

2. Le Traitement de texte attend la donnée D2 bloquée par l'application SGBD qui attend la donnée D4 bloquée par l'application CAO qui attend la donnée D5 bloquée par le Tableur qui attend la donnée D1 bloquée par le Traitement de texte. Cette situation est un **interblocage**.



3. Route : A, B, E, F

4. Pour réaliser ce schéma, il suffit de repérer les passerelles identiques à la destination pour chaque routeur.



Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

Exercice 3 - Programmation, Tableaux - Calcul des soldes

1.a.

```
def total_hors_reduction(tab):
    """ Calcule le total du panier. """
    total = 0
    for article in tab:
        total += article
    return total
```

1.b.

```
def offre_bienvenue(tab):
    """ tableau -> float """
    somme = 0
    longueur = len(tab)
    if longueur > 0 :
        somme = tab[0] * 0.8
    if longueur > 1 :
        somme = somme + (somme * 0.7)
    if longueur > 2 :
        for i in range(2, longueur):
            somme = somme + tab[i]
    return somme
```

2.

```
def prix_solde(tab) :
    """ Calcule le total du panier avec les soldes."""
    total = total_hors_reduction(tab)
    if len(tab) >= 5:
        total *= 0.5
    else:
        total *= 1 - (0.1 * len(tab))
    return total
```

3. a.

Solution 1 :

```
def minimum(tab):
    """ Renvoie la valeur minimum."""
    val_min = tab[0]
    for val in tab:
        if val < val_min:
            val_min = val
    return val_min
```

Solution de facilité :

```
def minimum(tab):
    """ Renvoie la valeur minimum
    return min(tab)
```

Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

3. b.

```
def offre_bon_client(tab):
    """ Calcule le total à payer avec l'offre bon client."""
    if len(tab) >= 2:
        return total_hors_reduction(tab) - minimum(tab)
    elif len(tab) == 1:
        return tab[0]
    else:
        return 0
```

4. a. `tab = [30.5, 15.0, 20.0, 6.0, 5.0, 35.0, 10.5]`
Le prix avec la promotion de déstockage est : $122 - 15 - 5 = 102€$

4. b. `tab = [35.0, 30.5, 20.0, 15.0, 10.5, 6.0, 5.0]`
Le prix avec la promotion de déstockage est : $122 - 20 - 6 = 96€$

4. c. Le client doit trier ses articles du plus cher au moins cher.

Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

Exercice 4 - Arbre de compétition

1. a. La racine est "Lea".
Les feuilles sont "Marc", "Lea", "Claire", "Theo", "Marie", "Louis", "Anne" et "Kevin".

1. b.

```
def vainqueur(arb) :
    """ Renvoie le nom du vainqueur du tournoi. """
    return racine(arb)
```

1. c.

```
def finale(arb) :
    """ Renvoie les noms des deux compétiteurs finalistes."""
    return [racine(gauche(arb)), racine(droite(arb))]
```

2. a. Il faut écrire une fonction récursive :

```
def occurrences(arb, nom, total=0) :
    """ Donne le nombre d'occurrences du joueur. """
    if est_vide(arb):
        return 0
    else:
        if racine(arb) == nom :
            total = 1
        else :
            total = 0
        return total + occurrences(gauche(arb), nom, total) + occurrences(droit(arb), nom, total)
```

2. b.

```
def a_gagne(arb, nom):
    """ détermine si le joueur a gagné au moins un match dans la compétition """
    return occurrences(arb, nom) > 1
```

3. a. Le vainqueur de la compétition voit son nom apparaître dans la racine de l'arbre. Pour le vainqueur de la compétition, la valeur sera erronée car elle comptera une fois de trop son nom.

3. b. Code corrigé :

```
def nombre_matchs(arb, nom):
    """ arbre_competition, str -> int """
    total = occurrences(arb, nom)
    if racine(arb) == nom:
        return total - 1
    else:
        return total
```

Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

4. Code complété :

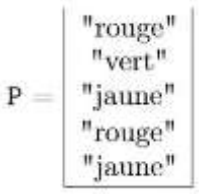
```
def liste_joueurs(arb):
    """ arbre_competition -> tableau """
    if est_vide(arb):
        return []
    elif est_vide(gauche(arb)) and est_vide(droit(arb)) :
        return [racine(arb)]
    else:
        return liste_joueurs(gauche(arb)) + liste_joueurs(droit(arb))
```

Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

Exercice 5 - Programmation, Piles, Files

1. a. La file F sera vide tandis que la pile P contiendra : “jaune”, “rouge”, “jaune”, “vert” et “rouge” (du fond vers le sommet de la pile).



1. b.

Solution : Programmation Procédurale du TAD

```
def taille_file(F):
    """ File -> Int """
    total = 0
    F_2 = creer_file_vide()
    while not est_vide(F):
        total += 1
        enfiler(F_2, defiler(F))
    F = F_2  # restore la file
    return total
```

Solution : Programmation Orientée Objet du TAD

```
def taille_file(F):
    """ File -> Int """
    total = 0
    F_2 = creer_file_vide()
    while not F_2.est_vide():
        total += 1
        F_2.enfiler(F.defiler())
    F = F_2  # restore la file
    return total
```

2.

Solution : Programmation Procédurale du TAD

```
def former_pile(F):
    """ File -> Pile """
    P = creer_pile_vide()
    while not est_vide(F):
        empiler(P, (defiler(F))
    # Renversement de la pile
    P_2 = creer_pile_vide()
    while not est_vide(P_2):
        empiler(P_2, depiler(P))
    return P_2
```

Solution : Programmation Orientée Objet du TAD

```
def former_pile(F):
    """ File -> Pile """
    P = creer_pile_vide()
    while not F.est_vide():
        P.empiler(F.defiler())
    # Renversement de la pile
    P_2 = creer_pile_vide()
    while not P_2.est_vide():
        P_2.empiler(P.depiler())
    return P_2
```

Correction

NSI - 2021 Amérique (21-NSIJ1AN1)

3.

Solution : Programmation Procédurale du TAD

```
def nb_elements(F, elt):
    """ File, element -> Int """
    total = 0
    F_2  = creer_file_vide()
    while not est_vide(F):
        temp = defiler(F):
        if elt == temp:
            total += 1
        enfiler(F_2, temp)
    F = F_2  # restore la file
    return total
```

Solution : Programmation Orientée Objet du TAD

```
def nb_elements(F, elt):
    """ File, element -> Int """
    total = 0
    F_2  = creer_file_vide()
    while not F.est_vide():
        temp = F.defiler():
        if elt == temp:
            total += 1
        F_2.enfiler(temp)
    F = F_2  # restore la file
    return total
```

4.

```
def verifier_contenu(F, nb_rouge, nb_vert, nb_jaune):
    """ File, Int, Int, Int -> Bool """
    r, v, b = nb_element(F, 'rouge'), nb_element(F, 'vert'), nb_element(F, 'jaune')
    return r <= nb_rouge and v <= nb_vert and j <= nb_jaune
```