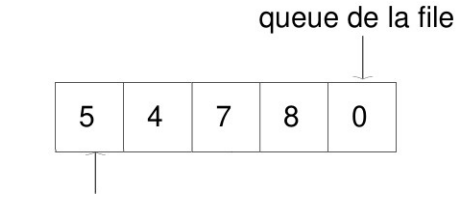
# Proposition de correction

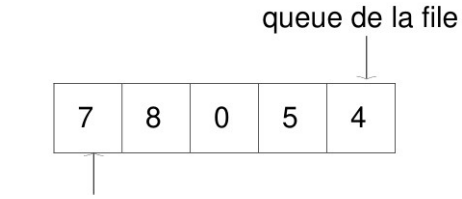
## Exercice 1

### Partie A

#### *Q1*




#### *Q2.a*




#### *Q2.b*

k = 5, taille de la file

### Partie B

#### *Q3*

```
def supprimer(f, e) :
    g = creer_file()
    while not est_vide(f) :
            x = defiler(f)
            if x != e :
                    enfiler(g, x)
    while not est_vide(g) :
            x = defiler(g)
            enfiler(f, x)
```

#### *Q4*

```
def placer_en_priorite(f, e) :
    g = creer_file()
    enfiler(g, e)
    while not est_vide(f) :
            x = defiler(f)
            if x != e :
```

```
                    enfiler(g, x)
    while not est_vide(g) :
        x = defiler(g)
        enfiler(f, x)
```

## Exercice2

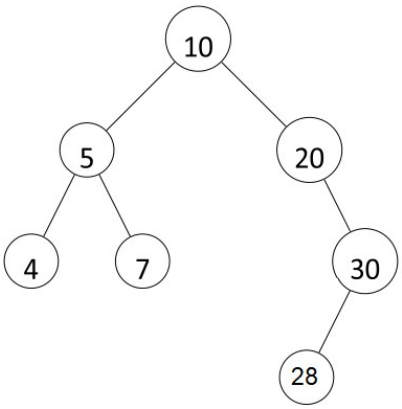
### Q1.a

4, 7, 5, 30, 20, 10

### Q1.b

infixe

### Q2.a




### Q2.b

4

### Q3.1

30

### Q3.2

La feuille de l'arbre sous droit

### Q4

```
def rechercher(a, e) :
    if est_vide(a) :
        return False
    elif racine(a) == e :
        return True
    elif e < racine(a) :
        return rechercher(sous_arbre_gauche(a), e)
```

```
    else :
        return rechercher(sous_arbre_droit(a), e)
```

## Exercice 3

### Q1

UPDATE produit

SET prix = 1.80

WHERE idProduit = 1

### Q2

La clé étrangère 'z' n'existe pas dans la table categorie.

La requête va générer une erreur.

### Q3.a

SELECT nomProduit

FROM produit

WHERE stock < 5

ORDER by nomProduit

### Q3.d

SELECT nomProduit

FROM produit

WHERE stock < 5 AND prix > 2

ORDER by nomProduit

### Q4

SELECT SUM(stock)

FROM produit

### Q5

SELECT produit.nomProduit, producteur.idProducteur, producteur.nomProducteur

FOM produit, producteur

WHERE producteur.bio = 1 AND producteur.idProducteur = produit.idP

ORDER BY  produit.nomProduit, producteur.nomProducteur

## Exercice 4

### Partie A

#### *Q1*

Carte réseau

#### *Q2*

switch

#### *Q3*

254

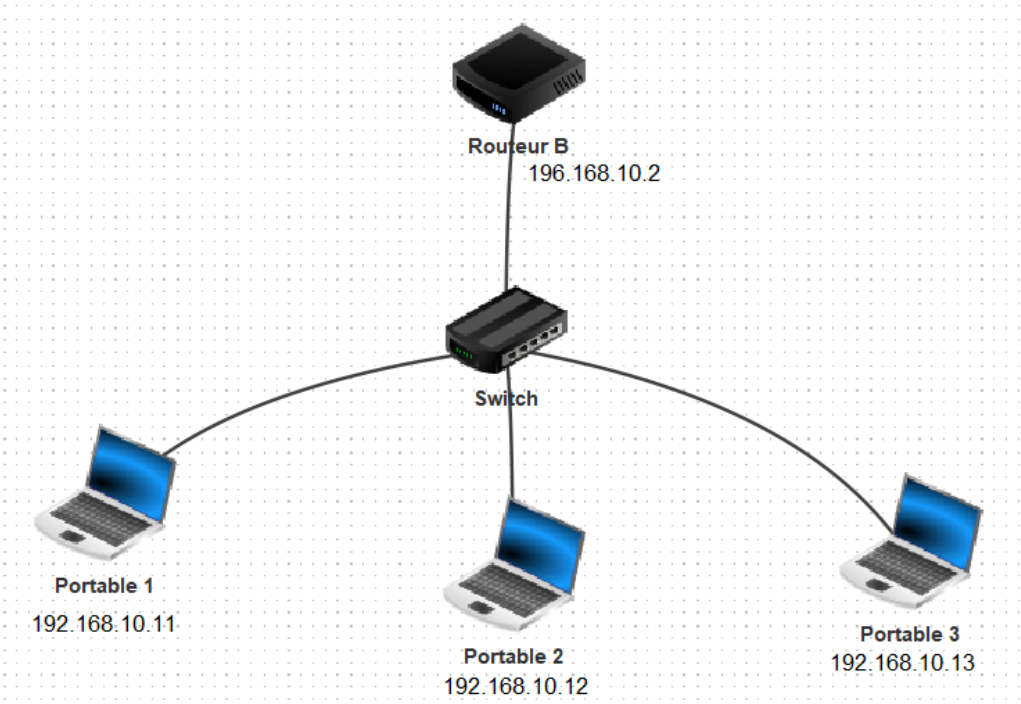
#### *Q4*

192.168.10.2

#### *Q5*

192.168.10.11

#### *Q6*



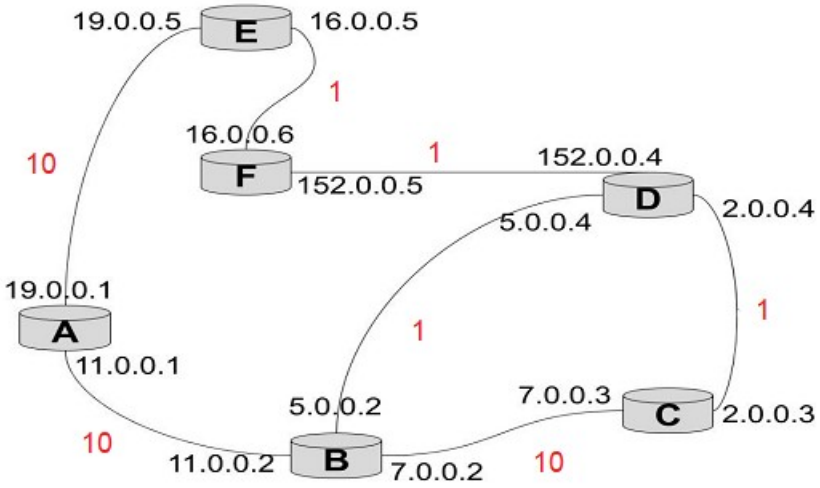

### Partie B

#### *Q7*

| Réseau de destination | Passerelle | Métrique |
|---|---|---|
| 2.0.0.0 | 7.0.0.2 | 1 |
| 5.0.0.0 | 5.0.0.2 | 0 |

| | | |
|---|---|---|
| 7.0.0.0 | 7.0.0.2 | 0 |
| 11.0.0.0 | 11.0.0.2 | 0 |
| 16.0.0.0 | 11.0.0.2 | 2 |
| 19.0.0.0 | 11.0.0.2 | 1 |
| 152.0.0.0 | 5.0.0.2 | 1 |

### *Q8*




B → D → F → E

coût = 1 + 1 + 1 = 3

### *Q9*

B → C → D → F → E

## Exercice 5

### Q1

Ligne 7 : mauvaise indentation

Ligne 8 : mauvaise indentation + manque opérateur d'addition

```
def calculer_score_sans_bonif(mot) :
    scrabble = {"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
                "H":4, "I":1, "J":8, "K":10,"L":1,"M":2,"N":1,"O":1,
                "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
                "X":10,"Y":10, "Z":10}
    score = 0
    for k in range(len(mot)) :
                score += scrabble[mot[k]]
    return score
```

## Q2

```
def calculer_score_sans_bonif(mot, bonif) :
    scrabble = {"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
                "H":4, "I":1, "J":8, "K":10,"L":1,"M":2,"N":1,"O":1,
                "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
                "X":10,"Y":10, "Z":10}
    score = 0
    for k in range(len(mot)) :
                score += scrabble[mot[k]] * bonif[k]
    return score
```

## Q3.a

```
def mot_lettre_position(liste : list, c : str, i : int) -> list:
    mots = []
    for l in liste:
        if i < len(l) and l[i] == c:
            mots.append(l)
    return mots
```

## Q3.b

```
mot_lettre_position(["BAL", "NOBLE", "ANE", "BALLE", "LOBE"], 'B', 2)
```

## Q4

```
def anagramme(mot1, mot2):
    if len(mot1) != len(mot2):
        return False
    elif len(mot1) == 0:
        return True
    elif not mot1[0] in mot2 :
        return False
    else :
        return anagramme(enlever(mot1, mot1[0]), enlever(mot2, mot1[0]))
```