Proposition de correction

Exercice 1

Q1

```
def plus_proche_voisin(t, cible):
    dmin = distance(t[0], cible)
    idx_ppv = 0
    n = len(t)
    for idx in range(1, n):
        dist = distance(t[idx], cible)
        if dmin < dist:
            dmin = dist
            idx_ppv = idx
    return idx_ppv
```

Q2

Linéaire

Q3.a

en stockant le résultat du calcul de distance() avant le test

Q3.b

de ne pas rechercher le plus grand élément de la liste à chaque fois ; il suffit de dépiler le premier élément.

Q3.c

```
def insertion(kppv : list, idx : int, distance : int):
    i= 0
    while i < len(kppv):
        index, dist = kppv[i]
        if dist < distance:
            break   # early exit
        i += 1
    kppv.insert(i, (idx, distance))
```

Exercice 2

Partie A

Q1

ifconfig

Q2

DHCP

Q3

192.168.1.1

Q4

C'est possible et cette adresse serait celle de la box vers Internet.

Q5

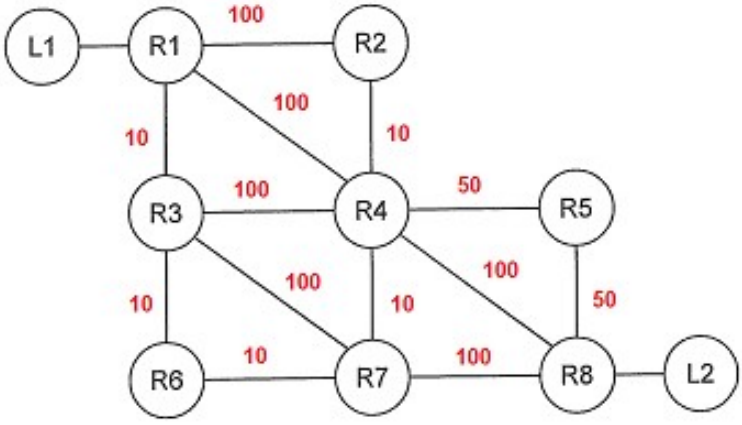
Oui, car les adresses 192.168.x.x ne sont pas routées sur Internet.

Partie B

Q1

$$C = \frac{10^9}{50 \times 10^6} = 20$$

Q2.a



Q2.b

R1-R3-R6-R7-R8

Q2.c

$$C_{max} = \frac{B_{ref}}{BP_{min}} < 30 \Leftrightarrow BP_{min} > \frac{B_{ref}}{30} = 33\ Mb/s$$

Exercice 3

Q1

```
UPDATE ModeleVelo
SET stock = 0
WHERE nomModele = 'Bovelo'
```

Q2

Requête 4

Requête 2

Q3.a

```
SELECT DISTINCTROW ModeleVelo.nomModele, Fabricant.nom
FROM ModeleVelo, Fabricant
WHERE ModeleVelo.stock = 0 AND ModeleVelo.idFabricant = Fabricant.idFabricant
ORDER BY ModeleVelo.nomModele, Fabricant.nom
```

Q3.b

```
SELECT COUNT(numeroCommande)
FROM Commande
WHERE DATE >= '2022-01-01'
```

Q3.c

```
SELECT DISTINCTROW Fabricant.nom
FROM ModeleVelo, Fabricant
WHERE ModeleVelo.stock > 0 AND ModeleVelo.idFabricant = Fabricant.idFabricant
ORDER BY Fabricant.nom
```

Q4

Le nom des clients qui ont commandé un modèle Bovelo

Exercice 4

Q1.a

```
from math import sqrt
```

Q1.b

```
def distance_points(a : tuple, b : tuple) -> real :
    return sqrt((b[0]**2 – a[0]**2) + (b[1]**2 – a[1]**2))
```

Q2

```
def distance(p : tuple, a : tuple, b : tuple) -> real :
    if a == b :
            raise ValueError('a et b identiques !')
    return distance_point_droite(p, a, b)
```

Q3

```
def le_plus_loin(ligne):
    n = len(ligne)
    deb = ligne [0]
    fin = ligne [n-l]
    dmax = 0
    indice_max = 0
    for idx in range(1, n-1):
        p = ligne[idx - 1]
        d = distance(p, deb, fin)
        if dmax < d:
            dmax = d
            indice_max = idx
    return indice_max, dmax
```

Q4

```
def extrait (tab, i, j):
    assert 0 <= i <= j < len (tab)
    return [tab[k] for k in range(i, j+1)]
```

Q5

```
def simplifie(ligne, seuil):
    n = len(ligne)
    if n <= 2:
            return ligne
```

```
    else:
            indice_max, dmax = le_plus_loin(ligne)

            if dmax <= seuil:
                    return [ligne[0], ligne[indice_max]]
            else:
                    return simplifie(extrait(ligne, 0, indice_max), seuil) + \
                            simplifie(extrait(ligne, indice_max, n-1), seuil)
```

Exercice 5

Q1

18

Q2.a

```
racine = Noeud(2)
gauche = Noeud(7)
racine.modifier_sag(gauche)
gauche.modifier_sag(Noeud(4))
gauche.modifier_sad(Noeud(1))
droite = Noeud(5)
racine.modifier_sad(droite)
droite.modifier_sag(Noeud(8))
```

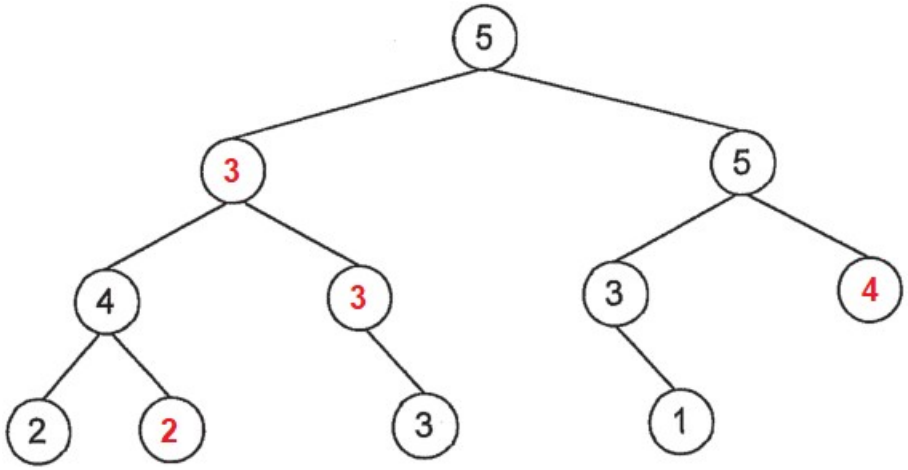
Q2.b

2

Q3

```
def pgde_somme(self):
    if self.sag != None and self.sad != None:
        return self.etiquette + max(self.sag.pgde_somme(), self.sad.pgde_somme())
    if self.sag != None:
        return self.etiquette + self.sag.pgde_somme()
    if self.sad != None:
        return self.etiquette + self.sad.pgde_somme()
    return self.etiquette
```

Q4.a



Q4.b

```
def est_magique(self):
    if self.sag.pgde_somme() != self.sad.pgde_somme():
        return False
    if self.sag != None and self.sad != None:
        return self.sag.est_magique() and self.sad.est_magique()
    return True
```