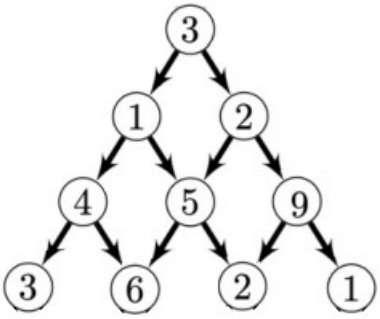
Correction

Exercice 1 : 6 points

Q1



0.5 point

Q2

3 → 2 → 5 → 6 (16)

0.25 point

Q3

2 → 5 → 2 (9)

2 → 5 → 3 (10)

2 → 1 → 3 (6)

2 → 1 → 9 (12)

0.25 point

Q4

Convention : racine n = 0

2^n

0.5 point

Q5

La complexité est exponentielle en $O(2^n)$.

0.25 point

Q6

```
def score_max(i : int, j : int, p : list) -> int:
    """
    @param i -- le niveau de la pyramide (1er niveau, i = 0)
    @param j -- indice de ligne sur le niveau
    @param p -- liste des éléments de la pyramide p
    @return le score de confiance
    """
    if i == len(p)-1:
        return p[len(p)-1][j]
```

1 point

```
    return p[i][j] + max(score_max(i+1, j, p), score_max(i+1, j+1, p))
```

Q7

```
def pyramide_nulle(n : int) -> list:
    """
    @brief    construit une pyramide remplie de 0 à n niveaux
    @param n -- le niveau max
    @return la pyramide
    """
    return [[0] * (i+1) for i in range(0, n+1)]
```

1 point

Q8

```
def prog_dyn(p):
    n = len(p)
    s = pyramide_nulle(n-1)
    # remplissage du dernier niveau
    for j in range(len(s[n-1])):
        s[n-1][j] = p[n-1][j]
    # remplissage des autres niveaux
    for i in range(n-2, -1, -1):
        for j in range(len(p[i])):
            s[i][j] = p[i][j] + max(s[i+1][j], s[i+1][j+1])
    # renvoie du score maximal
    return s[0][0]
```

1 point

Q9

Lignes 8 et 9 : 1 + 2 + 3 + … + (n+1) itérations = 1/2 n.(n+2) itérations

Soit une complexité en $O(n^2)$

0.25 point

Q10

Il faut mémoïser les calculs en cache.

```
def score_max_memo(i : int, j : int, p : list, cache : list) -> int:
    """
    @param i -- le niveau de la pyramide (1er niveau, i = 0)
    @param j -- indice de ligne sur le niveau
    @param p -- liste des éléments de la pyramide p
    @param cache -- mémoire cache pour mémoïsation
    @return le score de confiance
    """
    if i == len(p)-1:
        return p[len(p)-1][j]
    if cache[i][j] == 0:
        cache[i][j] = p[i][j] + max(score_max(i+1, j, p), score_max(i+1, j+1, p))
    return cache[i][j]
```

1 point

Exercice 2 — 6 points

Q1

- Logiciel Libre : donne 4 droits aux utilisateurs (utilisation, étude, modification, redistribution)
- Logiciel propriétaire : utilisation encadrée du logiciel (redevance), redistribution interdite

0.5 point

Q2

Interface entre les ressources et les logiciels de l'ordinateur. **0.5 point**

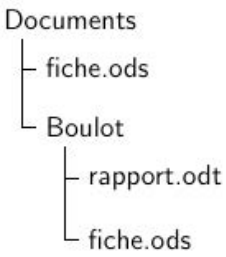
Q3

/home/elsa/documents/boulot/rapport.odt **0.5 point**

Q4

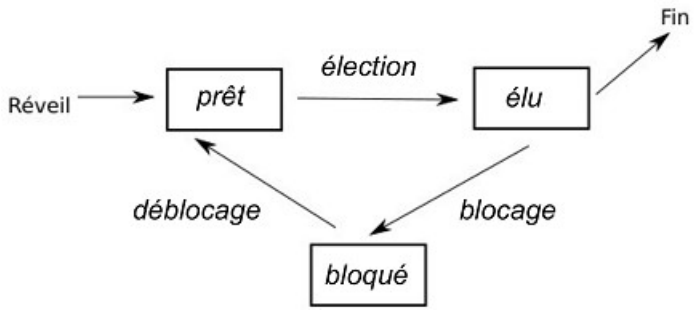
../max/images/photos_vac/photo_1.jpg **0.5 point**

Q5



0.5 point

Q6



0.5 point

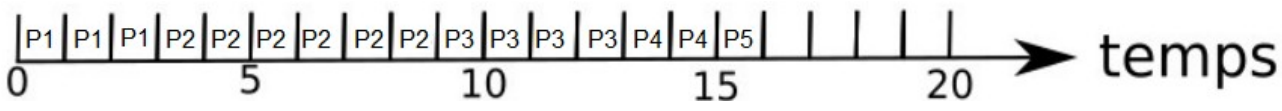
Q7

Attente d'une ressource (ex : imprimante) **0.25 point**

Q8

Une pile **0.25 point**

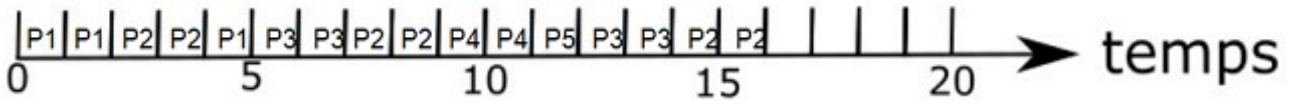
Q9



1 point

Q10

1 point



Q11

1. P1 prend R1
2. P2 prend R2
3. P1 demande R2
4. P2 demande R1

0.5 point

Exercice 3 — 8 points

Partie A

Q1

- a : [10, 8, 9, 9, 8, 10, 6, 7, 8, 8]
- b : Fondation

0.25 point

Q2

```
def titre_livre(dico, id_livre):
    for i in range(len(dico['id'])):
        if dico['id'][i] == id_livre :
            return dico['titre'][i]
    return None
```

0.5 point

Q3

```
def note_maxi(dico : dict) -> int:
    """
    @param dico -- dictionnaire de même structure que dico_livres
    @return la note maximale
    """
maxi = dico['note'][0]
    for note in dico['note']:
        if note > maxi :
            maxi = note
 return mini
```

0.5 point

Q4

```
def livres_note(dico : dict, note : int) -> list:
    """
    @param dico -- dictionnaire de même structure que dico_livres
    @param note -- une note [0 ; 10]
    @return la liste des titres des livres ayant obtenu la note n
    """
    return [dico['titre'][i] for i in range(len(dico['note'])) if dico['note'][i] == note]
```

0.5 point

Q5

```
def livre_note_maxi(dico : dict) -> list:
    """
    @param dico -- dictionnaire de même structure que dico_livres
    @return liste des titres des livres ayant obtenu la meilleure note
    """
    return livres_note(dico, note_maxi(dico))
```

0.5 point

Partie B

Q6

- Attribut : self.id
- Méthode : get_id()

0.25 point

Q7

```
def get_note(self):
    return self.note
```

0.25 point

Q8

```
biblio = Bibliotheque()
biblio.ajout_livre( Livre(5, "Blade Runner", "K.Dick", 1968, 8) )
assert biblio.titre_livre(5) == "Blade Runner"
```

0.25 point

Q9

```
def titre_livre(self, id_livre):
    for livre in self.liste_livre :
        if livre.get_id() == id_livre :
            return livre.get_titre()
    return None
```

0.75 point

Partie C

Q10

Un auteur peut avoir écrit plusieurs livres. La clef n'est donc pas unique.

0.25 point

Q11

Ubik

Blade Runner

0.25 point

Q12

```
SELECT titre
FROM livres
WHERE auteur = 'Asimov'
AND ann_pub > 1950
```

0.5 point

Q13

0.5 point

```
UPDATE livres
SET note = 10
WHERE id = 4
```

Q14

Évite de dupliquer les données et facilite la mise à jour. **0.5 point**

Q15

Clef étrangère vers la table auteurs pour la mise en relation des tables livres et auteurs. **0.5 point**

Q16

```
SELECT DISTINCT auteurs.nom, auteurs.prenom
    FROM auteurs
    JOIN livres ON livres.id_auteur = auteurs.id
    WHERE livres.ann_pub > 1960
```

0.75 point

Q17

0.5 point

Sélectionne les titres des livres dont leurs auteurs avaient moins de 30 ans au moment de leur publication.

Q18

Création d'une base de données, contenant des données personnelles, sujette à déclaration préalable auprès de la CNIL.

0.5 point