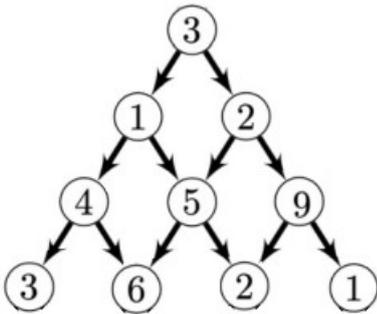


Proposition de correction

Exercice 1

Q1



Q2

$3 \rightarrow 2 \rightarrow 5 \rightarrow 6$ (16)

Q3

$2 \rightarrow 5 \rightarrow 2$ (9)

$2 \rightarrow 5 \rightarrow 3$ (10)

$2 \rightarrow 1 \rightarrow 3$ (6)

$2 \rightarrow 1 \rightarrow 9$ (12)

Q4

Convention : racine $n = 0$

2^n

Q5

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

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]
  
```

```
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)]
```

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]
```

Q9

Lignes 8 et 9 : $1 + 2 + 3 + \dots + (n+1)$ itérations = $\frac{1}{2} n \cdot (n+2)$ itérations

Soit une complexité en $O(n^2)$

Q10

Il faut mémoriser 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 mémorisation
    @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]
```

Exercice 2

Q1

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

Q2

Intergiciel entre les ressources et les logiciels de l'ordinateur.

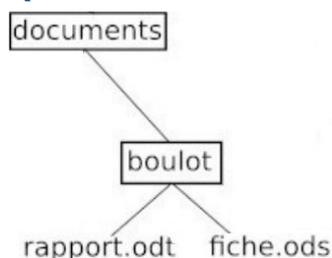
Q3

/home/elsa/documents/boulot/rapport.odt

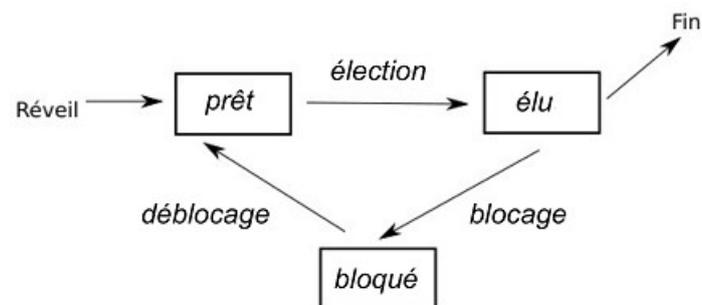
Q4

../max/images/photos_vac/photo_1.jpg

Q5



Q6



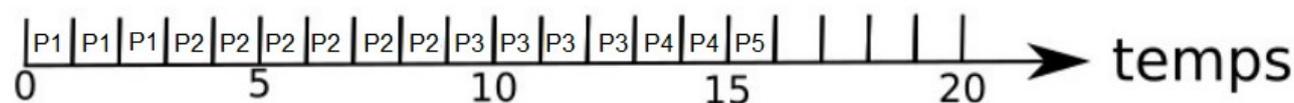
Q7

Attente d'une ressource (ex : imprimante)

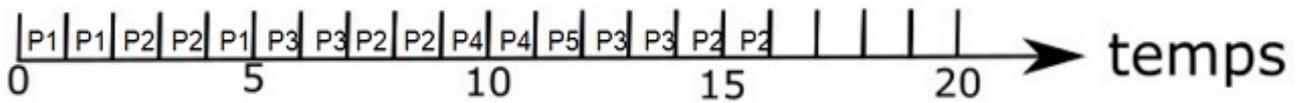
Q8

Une pile

Q9



Q10



Q11

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

Exercice 3

Partie A

Q1

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

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
```

Q3

```
def note_maxi(dico : dict) -> int:
    """
    @param dico -- dictionnaire de même structure que dico_livres
    @return la note maximale
    """
    mini = dico['note'][0] if len(dico) else None
    for note in dico['note']:
        if note < mini :
            note = mini
    return mini
```

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]
```

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))
```

Partie B

Q6

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

Q7

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

Q8

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

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
```

Partie C

Q10

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

Q11

Ubik

Blade Runner

Q12

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

Q13

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

Q14

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

Q15

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

Q16

```
SELECT DISTINCTROW auteurs.nom, auteurs.prenom  
FROM auteurs, livres  
WHERE livres.ann_pub > 1960  
AND livres.id_auteur = auteurs.id  
ORDER BY auteurs.nom, auteurs.prenom
```

Q17

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.