Proposition de correction

Exercice 1

Q1

1 + 3 = 4

Q2.a

elle s'appelle elle même en ligne 8

Q2.b

Si longueur de la chaîne < 2 alors fin du programme

sinon appel récursif de la chaîne diminuée de 2 caractères, donc longueur de chaîne → 0

Q3.a

TypeError

Q3.b

```
assert type(chaine) is str, "Erreur de type"
```

Q4

```
def est_palindrome(chaine : str) -> bool:
    n = len(chaine)
    for i in range(n // 2):
        if chaine[i] != chaine[n-i-1]:
            return False
    return True
```

Exercice 2

Q1.a

- num_cage : INT
- taille_cage : INT
- secteur_cage : VARCHAR(20)

Q1.b

num_reservation

Q1.c

- num_client
- num_animal
- num_cage

Q2.a

Api

Rex

Rex

Q2.b

SELECT client.nom_client

FROM client, reservation

WHERE reservation.num_cage = 23

AND client.num_client = reservation.num_client

ORDER BY client.nom_client

Q2.c

INSERT INTO animal

VALUES(492, ‘Suki’, ‘chat’, ‘petit’, 342)

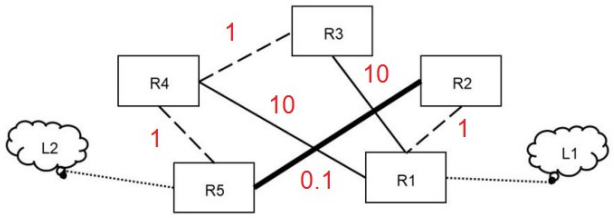
Q3.a

[26, 345]

Q3.b

```
def nombre_reservation(table, numero_client):
    """Paramètres :
    table : liste de dictionnaires, représentant les réservations
    numero_client : un entier, représentant le numéro du client concerné
    Valeur renvoyée : le nombre d'occurrences du numéro du client concerné. """
    occurrences = 0
    for ligne in table :
            if ligne[‘num_client’] == str(numero_client) :
                    occurrences += 1
    return occurrences
```

Q4.a



R1 → R2 → R5 : c = 1 + 0,1 = 1,1

Q4.b

R1 → R4 → R5 : c = 10 +1 = 11, coût minimal

Exercice 3

Q1

15 5 − 4 12 + × = (15 – 5) x (4 + 12) = 160

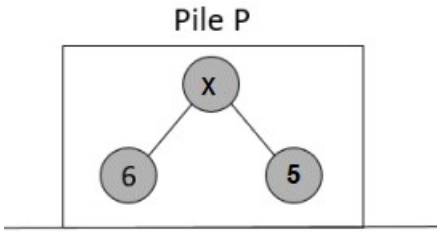
Q2

Postfixe

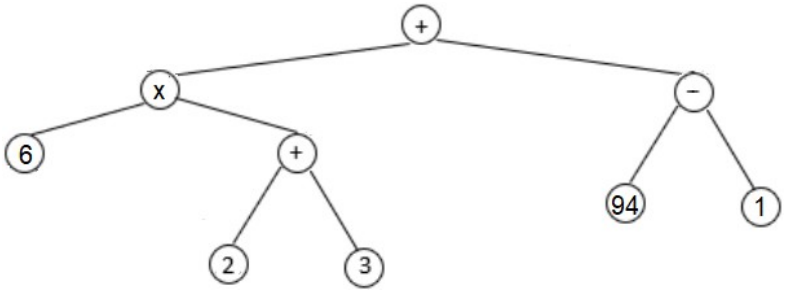
Q3.a

LIFO : Last In First Out

Q3.b



Q3.c



Q4

```
def evaluer(arb):
    if est_vide(gauche(arb)) and est_vide(droit(arb)):
            res = racine(arb)
    elif racine(arb) == "+":
            res = evaluer(gauche(arb)) + evaluer(droit(arb))
    elif racine(arb) == "-":
            res = evaluer(gauche(arb)) - evaluer(droit(arb))
    elif racine(arb) == "*":
            res = evaluer(gauche(arb)) * evaluer(droit(arb))
    else :
            res = evaluer(gauche(arb)) / evaluer(droit(arb))
    return res
```