Proposition de correction

Exercice 1

Q1

	Initialisation	Etape 1	Etape 2	Etape 3	Etape 4	...
i ≤ j		Vrai	Vrai	Vrai	Faux	
mot[i] ≠ mot[j]		Faux	Faux	Faux		
i	0	1	2	3	3	
j	4	3	2	1	1	
p	Vrai	Vrai	Vrai	Vrai	Vrai	

Q2.a

3

Q2.b

- n / 2 si n pair
- 1 + n // 2 si n impair

NB : il suffit de faire le test i < j pour avoir n /2 dans tous les cas

Q3

j – i = longueur(mot) – 1 – i

pour n = 0, i = 0, j – i = longueur(mot) – 1

pour n ≠ 0, j – n – i + n = longueur(mot) – 1 = cte

Q4

n = 7, donc 4 fois

la condition de continuation doit être (i < j) ET (p = Vrai)

Exercice 2

Q1.a

Relation plat	
Attribut	**Domaine**
id_plat	INT

nom_plat	VARCHAR(100)
type_plat	VARCHAR(100)
prix_plat	FLOAT

Q1.b

- plat(id_plat, nom_plat, type_plat, prix_plat)
- table_salle(num_table, nb_couvert_table, type_table)
- client(num_client, nom_client, prenom_client, date_naiss_client, mel_client, tel_client)
- reservation(num_reserv, nb_pers_reserv, date_reserv, num_table, num_client)

Q1.c

- reservation(num_reserv, nb_pers_reserv, date_reserv, #num_table, #num_client)

permet de faire le lien avec les autres tables (intégrité référentielle)

Q2.a

```
SELECT nom_plat, type_plat, prix_plat
FROM plat
ORDER BY nom_plat
```

Q2.b

```
SELECT nom_plat
FROM plat
WHERE type_plat = 'Dessert'
ORDER BY nom_plat
```

Q2.c

```
UPDATE client
SET tel_client = '0602030405'
WHERE num_client = 42
```

Q2.d

```
SELECT client.nom_client
FROM client, reservation
WHERE reservation.num_table = 13
AND reservation.num_client = client.num_client
ORDER BY nom_client
```

Exercice 3

Q1.a

cd ../projet

Q1.b

cd ~/projet

Q2.a

ls

Q2.b

chmod go-r config.txt

Q3.a

rm ne supprime que les dossiers vides

rm -R supprime récursivement les fichiers du dossier avant de supprimer le dossier

Q3.b

Parcours postfixe

Q4

le nombre de fichiers commençant par la lettre b, donc 1

Exercice 4

Q1

```
def ajouter_beurre(self, qt) :
    self.qt_beurre += qt
```

Q2

```
def afficher_stock(self) :
    print("farine:", self.qt_farine)
    print("oeuf:", self.qt_oeufs)
    print("beurre:", self.qt_beurre)
```

Q3

```
def stock_suffisant_brioche(self) :
    return self.qt_farine >= 350 and self.qt_oeufs >= 4 and self.qt_beurre >= 175
```

Q4.a

2

le nombre de brioches produites

Q4.b

farine: 300
oeuf: 2
beurre: 750

Q5

```
def nb_brioches(liste_stocks) :
    brioches = 0
    for s in liste_stocks :
            brioches += s.produire()
    return brioches
```

Exercice 5

Q1.a

[2, 6]

Q1.b

Le nombre de déplacements selon les axes x et y

Q2

```
def accessible(dep : str, arrivee : list) -> bool:
    return mystere(dep) == arrivee
```

Q3

```
from random import randint

def chemin(arrivee):
    """ arrivee : couple [x,y] avec x+y = 8 """
    deplacement = '00000000'
    while not accessible(deplacement, arrivee) :
        deplacement = ''
        for k in range(8):
            pas = str(randint(0,1))
            deplacement = deplacement + pas
    return deplacement
```

Q4

[5,3] → '11111000' = 248