1. **1.2.** À l'aide des annexes, trouver la requête exécutée par la fonction getRestoByIdR().

```
select * from resto where idR=:idR
```

1.3. Expliquer pourquoi la requête SQL ne retourne qu'une seule ligne.

parce que l'idR est la clé primaire de la table, et que par conséquent, sélectionner une ligne par son idR permet d'etre sur qu'il n'en ait qu'une seule.

1.4. Rappeler la syntaxe utilisée pour accéder à un champ d'un tableau associatif.
$tableau[indice]

1.5 Rappeler comment accéder au nom du restaurant dans la variable retournée par la fonction getRestoByIdR($idR).

getRestoByIdR($idR)[]

2.1. Quelle requête SQL est envoyée à la fonction prepare() dans getRestosByNomR() ?

```
select * from resto where nomR like :nomR
```

2.2. Quelle requête SQL est réellement exécutée après l'appel à bindValue() ?
select * from resto where nomR like %**contenu de $nomR***%

2.3. Cette requête est-elle susceptible de retourner plusieurs lignes ?
Oui, c'est meme le but, puisque l'utilisateur, s'il cherche un restaurant par nom, il est susceptible de ne pas connaitre le nom entier du restaurant.*

2.4. Parmi les 2 propositions suivantes une seule est vraie, justifier votre proposition à l'aide du code de la fonction présent en annexe 2, de vos connaissances en PHP et en SQL :

b) la variable `$resultat` retournée par la fonction `getRestosByNomR()` est un tableau dont chaque case est un tableau associatif. Cette variable peut contenir les informations sur plusieurs restaurants.

Je pense que c'est la B car le résultat de l'appel de `getRestosByNomR()` peut en effet contenir les informations d'un et d'un seul restaurant, si la requête SQL ne renvoie qu'un résultat, mais si celle-ci en renvoit plusieurs, la variable $resultat les prendra tous, et le moyen le plus efficace de prendre tous ces résultats est d'utiliser un tableau de tableaux.

3.1. Expliquer le rôle de la requête ci-dessous présente dans la fonction getNoteMoyenneByIdR()

```
select avg(note) from critiquer where idR=:idR
```

La requête renvoie une moyenne des notes données par les utilisateurs à un restaurant.

3.2. Combien de résultats sont attendus de la requête SQL ?

Un seul, le résultat de la commande avg

3.3. Comment est nommée la colonne affichée dans le résultat

avg(note)

Exécuter la requête en donnant une valeur impossible à la propriété idR. Par exemple 0 ou -1.

(ça renvoie NULL)

3.4. Quelle valeur est obtenue dans le cas où le calcul peut être fait ? Quelle valeur est obtenue dans le cas ou le calcul ne peut être fait ?

3,5 si le calcul est faisable, NULL sinon

3.5. À partir des questions précédentes, et en consultant le code de la fonction indiquer quel résultat sera retourné par la fonction lorsque l'identifiant d'un restaurant passé en paramètre existe ou n'existe pas dans la base de données.

Le resultat renvoyé par la fonction sera 0

3.6. Expliquer la condition suivante située à la fin de la fonction :

```
if ($resultat["avg(note)"] != NULL) {

return $resultat["avg(note)"];

} else {

return 0;

}
```

si le resultat de l'execution de la requête (le contenu de $resultat à l'indice avg(note)) n'est pas "NULL", on renvoie le resultat, sinon on renvoie 0

4.1. Expliquer le rôle de la requête présente dans la fonction addAimer().

```
insert into aimer (mailU, idR) values(:mailU,:idR)
```

C'est une requête permettant d'ajouter à la table aimer le restaurant dont l'idR est passé en paramètre ainsi que le mail de l'utilisateur qui l'a ajouter

4.2. Expliquer pourquoi les deux appels à bindValue() n'utilisent pas la même constante PDO en 3ème paramètre.

```
$req->bindValue(':idR', $idR, PDO::PARAM_INT);

$req->bindValue(':mailU', $mailU, PDO::PARAM_STR);
```

Pour faire appel à la base liée à l'utilisateur? Je devine que sans ça on appellerait toujours la même base alors que là puisque l'un des deux parametres est le mail de l'utilisateur ça me parait coherent que ça serve à ça.

4.3. A l'aide de l'annexe 11, déterminer quelle valeur est retournée par la fonction execute() en cas de réussite ou d'échec.

True en cas de réussite et False en cas d'echec

4.4. Déduire de la question précédente le type de données retourné par la fonction addAimer().

Un booleen

4.5. Si un utilisateur tente d'aimer un restaurant qu'il aime déjà, quelle sera la valeur retournée par la fonction addAimer() ?

False

5.1. Quelle est la clé primaire de la table aimer ?

idR et mailU

5.2. Rappeler la syntaxe de la requête de suppression en SQL.

delete from (la table) where (clef primaire de ce que l'on veut supprimer)

5.3. Écrire la requête SQL permettant de supprimer une occurrence précise de la table aimer.

delete from aimer where idR = (l'idR qu'on veut) and mailU = (mailU qu'on veut)

5.4. Se connecter à phpmyadmin et tester la requête à l'aide d'un exemple.

delete from aimer where idR = 11 AND mailU = "nicolas.harispe@gmail.com"

5.5. Dans quel script du modèle doit être placée la fonction delAimer() ? Justifier.

bd.aimer.inc.php (deja parce que c'est là qu'elle est dans la version finie) mais surtout parce que la fonction interagit avec la table aimer donc ça parait logique de la mettre avec les autres qui interagissent avec cette table.

5.6. Quels sont les paramètres à transmettre à la fonction delAimer() permettant de supprimer précisément une occurrence de la base de données ? En déduire le prototype de la fonction.

l'idR du Restaurant à retirer des restaurant aimer et le mail de l'utilisateur qui veut le retirer de ses restaurants aimés

delAimer($idR, $mailU) {

executer la requete delete from aimer where idR = $idR AND mailU = $mailU

retourner un truc pour savoir si la requête a bien été exécutée

}

5.7. Écrire le code de la fonction, puis ajouter un appel à cette fonction dans la section de test du script modèle approprié.

```
function delAimer($mailU, $idR) {

        try {

            $cnx = connexionPDO();

              $req = $cnx->prepare("delete from aimer where mailU=:mailU
and  idR=:idR");

            $req->bindValue(':idR', $idR, PDO::PARAM_INT);

            $req->bindValue(':mailU', $mailU, PDO::PARAM_STR);



            $req->execute();

            $resultat = $req->fetch(PDO::FETCH_ASSOC);

        } catch (PDOException $e) {

            print "Erreur !: " . $e->getMessage();

            die();

        }

        return $resultat;

    }
```

6.1. Écrire la définition (prototype) de la fonction getCritiquerByIdR().

getCritiquerByIdR($idR) {

executer la requete : select * from critiquer where idR = $idR

retourner un truc pour dire que la requete a été executée si elle l'a été sinon un truc pour dire que c'était pas le cas

}

6.2. Écrire puis tester la requête SQL permettant de récupérer les critiques associées à un restaurant.

select * from critiquer where idR = $idR

La fonction getCritiquerByIdr() doit renvoyer un résultat similaire à celui présenté dans l'annexe 13.

6.3. À l'aide de la section de code issue de la vue et de l'annexe 13 décrire la structure de données retournée par la fonction getCritiquerByIdR().

c'est un tableau de tableaux contenant les resultats de l'execution de la requete select * from critiquer where idR = 1

6.4. La fonction getCritiquerByIdr() actuellement présente dans le script modèle bd.critiquer.inc.php ne retourne qu'une liste vide. Compléter son code afin que celle-ci retourne la liste des critiques du restaurant dont l'identifiant est passé en paramètre.

Vérifier le bon fonctionnement de la fonction en ajoutant un appel dans la section de test du script bd.critiquer.inc.php puis en consultant la fiche d'un restaurant associé à des critiques.