```
    end
    --• Sinon, supprimer la séance
    delete  from …… where …….
END
```

4. On veut créer une autre version de la procédure utilisateur « sp_supp_seance », cette version force la suppression de la séance et ses absences. Compléter le code suivant de cette version de la procédure « sp_supp_seance »: (**2points**)

```
create PROCEDURE sp_supp_seance;2(@ids bigint)
AS
BEGIN
    -- supprimer les absences marquées dans cette séance,
    ……
    ……
    -- supprimer la séance
    ……
    ……
END
```

5. Lors de l'ajout ou de la modification d'une séance, elle ne doit pas se chevaucher avec d'autres séances actives (SUPPRIMER_LE est vide). Soit deux séances S1 et S2 :

	ID	MATIERE	CLASSE	JOUR	DE	A	CREER_LE	SUPPRIMER_LE	SALLE
S1 →	**1**	**<M1>**	**<C1>**	**<J1>**	**<DE1>**	**<A1>**		**VIDE**	
S2 →	**2**	**<M2>**	**<C2>**	**<J2>**	**<DE2>**	**<A2>**		**VIDE**	

Par définition, S1 et S2 se chevauchent si :

- elles sont de la même classe (C1=C2) **ou** du même professeur (M1 et M2 ont le même professeur),
- **et** dans le même jour (J1=J2),
- **et** leurs intervalles]DE,A[se chevauchent :]DE1,A1[∩]DE2,A2[≠Ø ⇔ (A1>DE2 et A2>DE1).

On suppose que nous avons créé une fonction « ***fn_SeancesChevaucheesAvec***» qui permet de retourner les séances chevauchées avec la séance dont les propriétés sont passées en paramètre :

```
CREATE FUNCTION
fn_SeancesChevaucheesAvec(
     @classe int,
     @matiere int,
     @jour int,
     @de time,
     @a time
)
RETURNS TABLE
AS
RETURN
(
     (
-- seances de la même classe
     SELECT * FROM
     fn_seancesMemeClasse(@classe)
     UNION -- ou
-- seances du même professeur
     SELECT * FROM
     fn_seancesMemeProf(@matiere)
     )
     INTERSECT -- et
-- seances du même jour
     SELECT * FROM
     fn_seancesMemeJour(@jour)
     INTERSECT -- et
-- seances dont ]DE,A[∩]@de,@a[≠Ø
     SELECT * FROM
     fn_SeancesInInterval(@de,@a)
)
```

a. Compléter le code des fonctions suivantes : (**2.5 points**)

`CREATE FUNCTION fn_SeancesMemeJour( @jour int ) RETURNS TABLE AS RETURN ( SELECT * from …… where jour=…… and …… is null )`	`CREATE FUNCTION fn_SeancesInInterval( @de time, @a time ) RETURNS TABLE AS RETURN ( SELECT * from seance where …… and …… and …… is null )`
Retourne les séances actives dont le jour est ***@jour***	Retourne les séances actives qui se chevauchent avec l'intervalle***]@de,@a[***
`CREATE FUNCTION fn_SeancesMemeProf( @matiere int ) RETURNS TABLE AS RETURN ( SELECT * from seance where matiere in ( select …… from fn_matieresMemeProf(……) ) and supprimer_le is null )`	`CREATE FUNCTION fn_matieresMemeProf( @matiere int ) RETURNS TABLE AS RETURN ( select * from matiere where professeur = ( select …… from matiere where id=…… ) )`
Retourne les séances du professeur dont la matière est @matiere	Retourne les matières enseignées par le professeur de ***@matiere***

b. Compléter le code suivant du déclencheur « tr_ajoutSeance » qui oblige l'insertion simple des séances et contrôle le problème de chevauchement entre les séances décrit précédemment dans (a) (**2.5 points**)

```
CREATE TRIGGER tr_ajoutSeance ON ……  FOR …… AS
BEGIN
    -- si nombre de lignes inserés > 1
    -- alors annuler la transaction en cours.
    --(utiliser la variable globale @@RowCount)
    if(……)
        ……
    -- à ce niveau @@RowCount=1
    declare @classe int,@matiere int,@jour int,@de time,@a time
    --recuperer les valeurs de ces variables depuis 'Inserted'
    select ……
    ……
    from inserted
```