# TP PROLOG
# Remarques et Exemple de test

**Chargé de cours :** Wajdi DHIFLI

**Remarques générales :**

- Lisez tout l'énoncé du TP
- Ne créez pas des prédicats autres que ceux demandés, sinon la réponse sera considérée fausse même si le prédicat fonctionne.
- Les prédicats qui ne respectent pas la syntaxe indiquée (surtout nom et nombre de paramètres) vont être considérés comme faux et ne vont pas être corrigés.
- Aucune soumission après la date limite ne sera acceptée.
- Il faut impérativement soumettre sur Moodle, les soumissions par email ne seront pas considérées.
- Le graph est non orienté : lien(x,y)= lien(y,x). Dans la représentation du graphe on ne représente pas le même lien dans les deux sens (faits lien(x,y) et faits lien(y,x)), mais plutôt on prend en compte le fait de la commutativité des liens dans le code des prédicats.

**Remarques par question :**

**Q1-** Avant la création d'un nœud dans la base des faits, il faut s'assurer qu'il n'existe pas déjà. Dans le cas contraire, il faut afficher un message d'information, et redemander la saisie de l'index. Le suivant est un exemple illustratif :

```
?- ajouter_noeud.
Donnez la liste des noeuds
Donnez l'index du noeud : 1.
noeud deja existant
Donnez l'index du noeud :
```

Après la saisie du nœud, le programme demande *« Voulez-vous continuer (oui/non)? »*. Le traitement sera comme suit, si la réponse est « oui », le programme doit boucler et demander l'index du nœud à ajouter, si la réponse est « non » ou n'importe quel autres saisies, le programme sort et s'arrête.

**Q2-** Il faut vérifier si le nœud existe déjà ou pas dans la base. S'il n'existe pas, rien n'est effacé et afficher un message *« noeud introuvable »*. À la suppression du nœud, tous les liens associés au noeud supprimé doivent être également supprimés.

**Q3-** Même consignes que Q1 plus : il faut s'assurer que :

- Après la saisie de chaque index si un nœud avec l'index indiqué existe ou pas sinon afficher un message *« noeud introuvable »* et recommencer toute la procédure
- Après la saisie de deux index existants, il faut verifier si le lien existe déjà dans la base sinon afficher un message *« lien deja existant »* et recommencer toute la procédure de saisie.

Sinon insérer le lien dans la base des faits et demander *« Voulez-vous continuer (oui/non)? »*. Le traitement sera comme suit, si la réponse est « oui », le programme doit boucler et recommencer par demander le premier index, si la réponse est « non » ou n'importe quel autres saisies, le programme sort et s'arrête.

**Q4-** Il faut vérifier si le lien existe déjà ou pas dans la base. S'il n'existe pas, rien n'est effacé et afficher un message *« lien introuvable »*. Sinon, supprimer le lien correspondant et afficher « lien(Index1,Index2) supprimé ».

**Q5-** Deux nœuds adjacents dans le graphe ne peuvent pas avoir une même étiquette. Il n'existe aucun lien entre deux nœuds qui ont le même label.

**Q6-** Le prédicat retourne « true » s'il existe un lien d'amitié : que ce soit direct (exemple : lien(x,y) : amitie(label_x, label_y)➔true aussi amitie(label_y, label_x)➔true) ou à travers d'autres amis intermédiaires (exemple : lien(x,y), lien(y,z), lien(z,a) : amitie(label_x, label_a)➔true, amitie(label_a, label_x)➔true).

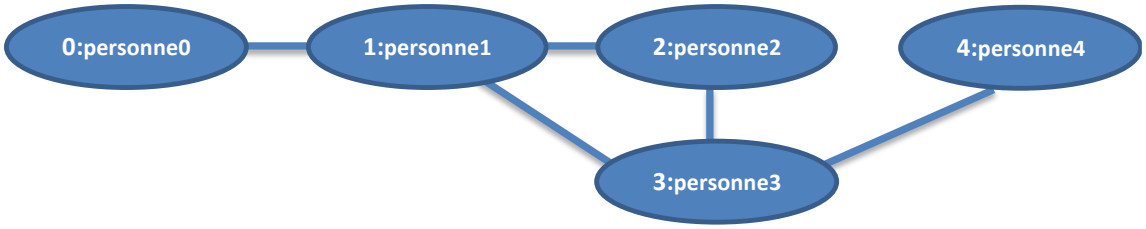
**Q7-** Le prédicat retourne dans Z les noms de tous les amis **directs** communs entre deux personnes (exemple : lien(x,z), lien(y,z) : commun(label_x, label_y, X)➔X=label_z ( ; les autres valeurs possibles de X s'il y a d'autres amis commun directs) aussi pour le meme exemple commun(label_y, label_x, X) donnera le même résultat.

**Q8-** Le prédicat « intermediaires(X,Y,N) » retourne dans N le nombre d'amis intermédiaires entre deux personnes (exemple lien(x,y), lien(y,z), lien(z,a) : intermediaires(label_x, label_z, N)➔N=1. Aussi intermediaires(label_x, label_a, N)➔N=2)

**Q9-** Le prédicat suggestion(X,Y) permet de suggérer des amis à la personne en paramètre. suggestion(label_x,Y) retourne dans Y (ou inversement suggestion(X, label_y) retourne dans X). Ce predicat utilise le predicat « intermediaires » pour compter le nombre d'intermédiaires. Si ce nombre est >=1 (c'est-à-dire amis de mon amis) ou < 3 (amis de l'amis de mon amis) autrement dit le nombre d'intermediaires est dans [1,2]. (exemple : lien(x,y), lien(y,z), lien(z,a), lien(a,b) : suggestion(label_x, Y)➔Y=label_z ; Y=label_a)

## Exemple :

Chaque nœud dans ce graphe est étiqueté par (**index du noeud : label du noeud**)




Les nœuds et les liens qui representent ce graphe sont :

noeud(0,personne0).
noeud(1,personne1).
noeud(2,personne2).
noeud(3,personne3).
noeud(4,personne4).
lien(0,1).
lien(1,2).
lien(1,3).
lien(2,3).
lien(3,4).

Résultats attendus :

**Q5-** *?- verifier.*

*graphe valide*

*true.*

Si on change le label du nœud 1 en personne0 (noeud(1,personne0)). Dans ce cas, le nœuds 0 et 1 sont liés et ont le même label. Dans ce cas :

*verifier.*

*graphe non valide*

*false*

**Q6-** *?- amitie(personne0,personne3).*

*true.*

*?- amitie(personne3,personne0).*

*true.*

**Q7-** *?- intermediaires(personne0,personne1,X).*

*X = 0 ;*

*false.*

*?- intermediaires(personne0,personne2,X).*

*X = 1 ;*

*false.*

*?- intermediaires(personne0,personne3,X).*

*X = 1 ;*

*X = 2 ;*

*false.*

← il existe deux solutions possibles car il y a deux chemins differents dans le graphe qui

*?- intermediaires(personne0,personne4,X).*

*X = 3 ;*

*X = 2 ;*

*false.*

*?- intermediaires(personne1,personne3,X).*

*X = 0 ;*

*X = 1 ;*

*false.*

**Q8-** *?- suggestion(personne0,X).*

*X = personne2 ;*

*X = personne3 ;*

*X = personne3 ;*

*X = personne4 ;*

← personne3 est proposé deux fois comme solution car on peut l'atteindre en passant par deux chemin différents dans le graphe. Dans les deux cas personne3 représente une solution possible pour personne0.

*false.*

*?- suggestion(personne1,X).*

*X = personne4 ;*

*X = personne4 ;*

*false.*

*?- suggestion(personne2,X).*

*X = personne4 ;*

*X = personne0 ;*

*false.*

*?- suggestion(personne3,X).*

*X = personne0 ;*

*X = personne0 ;*

*false.*

*?- suggestion(personne4,X).*

*X = personne0 ;*

*X = personne1 ;*

*X = personne1 ;*

*X = personne2 ;*

*false.*

*?- suggestion(Y,personne2).*

*Y = personne0 ;*

*Y = personne4 ;*

*false.*

*?- suggestion(personne0,personne2).*

*true .*