TP PROLOG

Chargé de cours : Wajdi DHIFLI

On veut créer un PROLOG qui réalise des fonctionnalités pour une application d'un réseau social. On va créer un graphe d'amis avec les contraintes suivantes :

1. Chaque personne est représentée par un nœud dans le graphe « ***noeud(Index, Label)*** »
2. Le graphe est étiqueté avec un nom d'une personne sur chaque nœud
3. Deux nœuds adjacents dans le graphe ***ne peuvent pas*** avoir une même étiquette

Créez un programme prolog qui permet de :

1. Créez le prédicat « ajouter_noeud » qui permet à l'utilisateur de saisir la liste des nœuds du graph. Au lancement le prédicat affiche « Donnez la liste des nœuds », puis d'une façon **répétitive**, il demande « donnez l'index du nœud : » et l'utilisateur doit saisir l'index du nœud. Le prédicat lit l'index puis demande « Donnez le nom de la personne : » après lecture du nom de la personne le prédicat ajoute le nœud dans la base des faits sous la forme ***noeud(Index, Label)***. Par la suite, le prédicat demande « Voulez-vous continuer (oui/non)? » si la réponse saisie est « non » le prédicat s'arrête. Si la réponse est « oui », le prédicat continue la boucle et redemande l'index du nouveau nœud. (2pts)
2. Créez le prédicat « supprimer_noeud » qui permet de supprimer un nœud de la base des faits. Le prédicat demande l'index du nœud à supprimer, l'utilisateur saisi l'index, puis le programme supprime le nœud correspondant et affiche « nœud(Index,Label) supprimé » (il faut afficher l'Index et le Label correspondants). Sinon il affiche « noeud introuvable ». (2pts)
3. Créez le prédicat « ajouter_lien » qui permet de saisir les liens entre les nœuds du graph, au lancement le prédicat affiche « Donnez la liste des liens », puis d'une façon **répétitive**, il demande « donnez l'index du premier nœud : » et l'utilisateur doit saisir l'index du premier nœud dans le lien. Par la suite, il fait similairement pour le deuxième nœud. Après lecture des indexes le prédicat ajoute le lien dans la base des faits sous la forme ***lien(Index1, Index2)***. Par la suite, le prédicat demande « Voulez-vous continuer (oui/non)? » si la réponse saisie est « non » le prédicat s'arrête. Si la réponse est « oui », le prédicat continue la boucle et redemande les nouveaux indexes des nœuds. (2pts)
4. Créez le prédicat « supprimer_lien » qui permet de supprimer un lien de la base des faits. Le prédicat demande les indexes (l'index du nœud1, il le lit puis il demande celui du noeud2 et il le lit) du lien à supprimer, l'utilisateur saisi les indexes, puis le programme supprime le lien correspondant et affiche « lien(*Index1*,*Index2*) supprimé **»** (il faut afficher les indexes correspondants)**.** Sinon il affiche « lien introuvable ». (2pts)
5. Créez le prédicat « verifier » qui permet de vérifier la validité du graphe selon les contraintes données au début (il est interdit qu'une personne ait un lien d'amitié direct

avec une autre personne ayant le même nom). Le prédicat affiche « graphe valide » si le graph respectent les contraintes sinon il affiche « graphe non valide ». (2,5pts)

6. Créez le prédicat « amitie » qui prend en paramètre les noms de deux personnes et retourne « true » si les deux personnes en paramètres ont un lien d'amitié : que ce soit direct ou à travers d'autres amis intermédiaires. (2,5pts)
7. Créez le prédicat « commun » (commun(X,Y,Z)) qui prend en paramètre deux noms (X,Y) et retourne dans son troisième paramètre (Z) les noms de tous les amis directs communs entre deux personnes. (exemple x1 amis avec x2, x2 avec x3, la requête commun(x1,x3,Z) donne Z=x2.) (2pts)
8. Créez le prédicat « intermediaires » (intermediaires(X,Y,N)) qui permet de compter (dans N) le nombre d'amis intermédiaires entre deux personnes (X et Y). (2,5pts)
9. Créez le prédicat « suggestion » qui permet de suggérer des amis à la personne en paramètre. Ce prédicat utilise le prédicat « intermediaires » pour déterminer le nombre d'amis intermédiaires (la distance). Les amis suggérés sont ceux ayant une distance >1 et <= 3. (exemple x1 amis avec x2, x2 avec x3, la requête suggestion(x1,Y) donne Y=x3.) (2pts)

Pour des raisons de tests et de conformité, les prédicats à développer doivent obligatoirement être dans les formats suivants :

1. ajouter_noeud (sans paramètres)
2. supprimer_noeud (sans paramètres)
3. ajouter_lien (sans paramètres)
4. supprimer_lien (sans paramètres)
5. verifier (sans paramètres)
6. amitie(X,Y)
7. commun(X,Y,Z)
8. intermediaires(X,Y,N)
9. suggestion(X,Y) (doit utiliser le prédicat « intermediaires »)

NB : Les prédicats qui ne respectent pas la syntaxe indiquée vont être considérés comme faux et ne vont pas être corrigés. N'utilisez que les prédicats mentionnés (il ne faut pas déclarer, ni utiliser des prédicats autres que ceux mentionnés).

Les travaux pratiques peuvent être réalisés individuellement ou en équipe de deux étudiants du même groupe au maximum. Le TP doit être réalisé dans un fichier contenant dans les deux premières lignes (en commentaire) les noms et codes permanents des co-équipiers.

Votre code sera vérifié d'une façon **AUTOMATIQUE**, vous serez donc pénalisés s'il ne correspond pas **EXACTEMENT** à ce qui est demandé.

Votre fichier doit pouvoir s'exécuter **SANS MODIFICATION** sur le système SWI-Prolog.

Votre TP doit être remis de façon électronique sur moodle selon la date prévue.

Vous devez remettre aussi une documentation (**1 page au maximum** dans le format PDF) qui explique le code réalisé.