Exercices sur les listes

Question 1 :

Ecrire la règle **liste_nombres_négatifs(L,R)** qui prend en entrée une liste ***L*** et qui retourne une liste ***R*** contenant tous les nombres négatifs de ***L***. Par exemple, la requête
?- liste_nombres_négatifs([10,-2,9, -7, 0, 3, 17, -1], R). doit retourner **R = [-2, -7, -1].**

Question 2 :

Ecrire la règle **supprimer_prédécesseurs(X, L, R)** qui permet de supprimer tous les prédécesseurs de l'élément ***X*** dans la liste ***L***, la liste ***R*** est le résultat de cette suppression. Par exemple, la requête **?- supprimer_prédécesseurs(7, [10, 2, 9, 7, 0, 3, 17], R).** doit retourner **R = [7, 0, 3, 17]**, et **?-supprimer_prédécesseurs(5, [10, 2, 9, 7, 0, 3, 17], R).** doit retourner **R = [].**

Question 3 :

Ecrire la règle **chercher(X, L)** qui permet de vérifier si l'élément ***X*** existe dans la liste ***L*** sachant que cette liste est triée dans un ordre croissant. La règle doit afficher un message pour dire si l'élément existe ou pas. Par exemple, la requête **?- chercher(7, [3, 5, 7, 10]).** doit afficher ***« L'élément 7 existe »*.**

Remarque : La solution générale, qui permet de chercher si l'élément appartient à une liste quelconque (triée ou pas), ne sera pas acceptée. ***Il faudra impérativement considérer le fait que la liste soit triée***.

Question 4 :

Ecrire la règle **insérer(X, I, L, R)** qui permet d'insérer l'élément ***X*** à la **I**ème position de la liste ***L***, la liste ***R*** est le résultat de cette insertion. Par exemple, la requête **?- insérer(12, 2, [3, 6, 1], R).** doit retourner **R=[3, 12, 6, 1]**, la requête **?- insérer(12, 4, [3, 6, 1], R).** doit retourner **R=[3, 6, 1, 12]**, et finalement la requête **?- insérer(12, 5, [3, 6, 1], R).** doit afficher le message d'erreur : « ***Erreur !!! La position demandée est incorrecte*** ». Le même message d'erreur doit être affiché si le **I** est inférieur ou égale à 0.

Question 5 :

Ecrire la règle **somme_prédécesseurs(X, L, S)** qui permet de calculer la somme ***S*** de tous les prédécesseurs de l'élément ***X*** dans la liste ***L***. Par exemple, la requête
?- somme_prédécesseurs(7, [10, 2,9, 7, 0, 3, 17], S). doit retourner **S = 21**, et
?- somme_prédécesseurs(5, [10, 2,6], S). doit retourner **S = 0.**

Question 6 :

Ecrire la règle **ajouter(X, L, R)** qui permet d'ajouter l'élément ***X*** à la liste ***L*** sachant que cette liste est triée dans un ordre croissant, la liste ***R*** est le résultat de cet ajout. Par exemple, la requête **?- ajouter(6, [3,5, 7, 10], R).** doit retourner **R=[3,5,6,7,10]**, **?- ajouter(-1, [-3, 5], R).** doit retourner **R=[-3,-1,5],** et **?-ajouter(16, [7, 10]).** doit retourner **R=[7,10,16].**

Question 7 :

Ecrire la règle **vérifier(I, X, L)** qui permet de vérifier si le **I**ème élément de la liste ***L*** est bien l'élément ***X***.

Par exemple, la requête **?- vérifier(2, 6, [3, 6, 1]).** doit afficher : « ***Il s'agit bien du 6*** ». la requête **?-vérifier(2, 4, [3, 6, 1]).** doit afficher : « ***Il ne s'agit pas du 4*** ».