Codage en parties communes

LE PROBLEME. On nous demande de réaliser un programme pour gérer une base de données décrivant une collection d'individus. Cinq informations sont associées à chacun d'eux :

- le sexe 1 ou 2
- l'année de naissance 0 à 99
- le mois de naissance 1 à 12
- le département de naissance 1 à 99
- le numéro d'inscription 1 à 700

Il doit être possible de réaliser les opérations suivantes :

- Insertion d'un nouvel individu dans la base. A cet effet, les cinq informations associées devront être toutes précisées. Si l'individu en question a déjà été enregistré, on refusera l'insertion (un message indiquera ce fait).
- Recherche d'informations. En réponse à une question indiquant la valeur de certaines des cinq informations, le programme devra énumérer tous les individus qui « s'unifient » avec la requête, c'est-à-dire pour lesquels celles des valeurs qui ont été précisées dans la question coïncident avec les informations correspondantes chez les individus.

Par exemple, si la base de données contient les informations

sexe	*année*	*mois*	*département*	*numéro*
1	68	12	04	209
2	70	06	75	365
1	49	07	83	130
2	72	02	04	520

alors on peut ajouter un nouvel élément défini par les cinq informations :

2	69	06	20	415

et on peut ensuite demander *quelles sont les filles nées au mois de juin* par la question

2	*	06	*	*

à laquelle le programme répond en affichant les deux seuls individus qui satisfont à la requête :

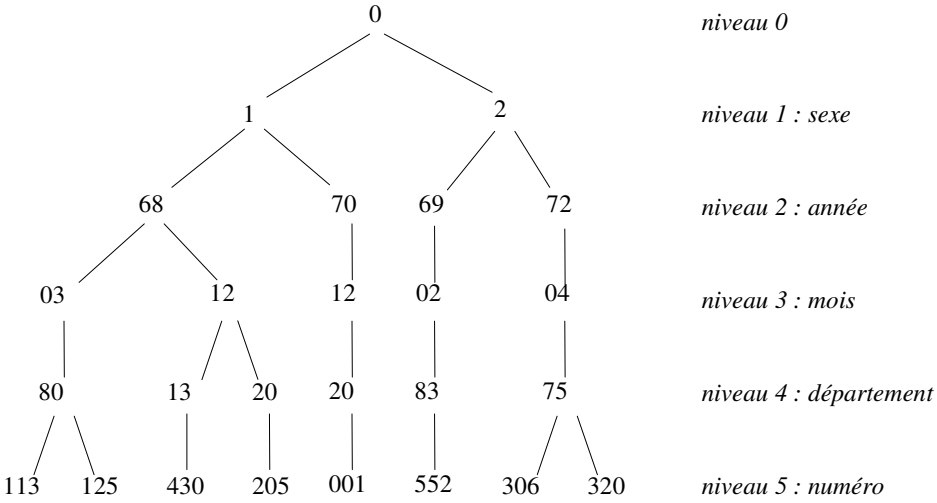
2	70	06	75	365
2	69	06	20	415

SUR LE CODAGE DES DONNEES. Les données seront regroupées dans un arbre dont chaque nœud porte une des cinq informations en question (dans un champ *info* de type entier). La racine, nœud de niveau zéro, sert d'accès général à toutes les branches mais ne contient aucune valeur significative dans son champ *info*.

Chaque *branche* (chemin allant de la racine à une feuille) de l'arbre comporte les cinq informations relatives à un individu donné : au niveau 1 le sexe, au niveau 2 l'année de naissance, et ainsi de suite. Au début l'arbre est réduit au seul nœud racine. L'introduction d'un nouvel individu se fait en recherchant dans l'arbre déjà mémorisé une branche commençant par les mêmes informations que les premières informations de l'individu nouveau, et en créant seulement les enregistrements à partir d'une information distincte. De cette manière, lorsque les quintuplets de valeurs définissant deux individus débutent par les mêmes valeurs, les informations communes sont codées en un seul exemplaire, d'où le nom de cette structure de données.

Enfin, pour optimiser les recherches, on range par ordre croissant les nœuds fils d'un nœud donné.

Exemple :



Chaque feuille (nœuds portant 113, 125, 430, 205, etc.) de cet arbre détermine une branche (chemin unique joignant la racine à cette feuille) qui définit un individu de la collection.

INTERFACE UTILISATEUR. Pour réaliser les deux opérations expliquées et pouvoir arrêter la consultation, le programme doit reconnaître et exécuter trois commandes (qu'on peut, par exemple, associer des nombres ou des lettres) :

inserer	insertion d'un nouvel élément
chercher	recherche d'un étudiant ou d'une collection d'étudiants
terminer	arrêt du programme

Lors d'une insertion, le programme demande à l'utilisateur cinq valeurs définissant un individu à insérer. Dans le cas d'une recherche, le programme demande également cinq valeurs, parmi lesquelles un signe conventionnel, par exemple '*', peut indiquer une valeur indifférente. La requête « *quelles sont les filles nées au mois de juin* » s'écrit donc « 2 * 6 * * ».

PROGRAMME A REALISER. Outre les fonctions auxiliaires qui vous paraîtront utiles, vous écrirez une fonction *chercher* qui parcourt l'arbre, guidée par les valeurs données par l'utilisateur. Le travail de cette fonction peut être défini de la manière suivante :

- pour tout nœud qui n'est pas une feuille, parcourir ceux de ses sous-arbres dont la racine porte une information compatible avec la valeur correspondante de la requête (l'ordre entre les frères permet d'arrêter ce parcours avant la fin des sous-arbres). Lorsqu'une des informations de la requête possède la valeur passe-partout, tous les sous-arbres doivent être examinés.
- pour un nœud qui est une feuille, imprimer les informations portées par la branche correspondante. Pour cela, ces informations auront été mémorisées dans une pile lors de la descente.

Vous écrirez de même une fonction *insérer* qui, étant données cinq « vraies » informations, recherche ou crée les nœuds correspondants, en respectant le principe du codage en parties communes et l'ordre entre les frères. Comme on l'a dit, l'insertion d'un individu qui se trouve déjà dans la banque doit être refusée.

Finalement, vous écrirez une fonction *dialogue* chargée de converser avec l'utilisateur, de lire ses requêtes et de lui fournir les réponses, et un programme principal pour tester tout cela.

☺☺