[LC580]

TP C : L'Arbre aux Animaux

```
Bonjour ! Pensez à un animal, et répondez à mes questions par OUI ou par NON (O/N) :
Est ce un mammifère ? OUI
Est ce un ruminant ? NON
A t'il des griffes rétractiles ? OUI

Je pense que c'est un chat.
 C'est bien un chat n'est ce pas ? OUI

Wouaouh!! J'ai trouvé votre animal!...
         Mais ce n'était pas trop dur!..
...
Bonjour ! Pensez à un animal, et répondez à mes questions par OUI ou par NON (O/N) :
Est ce un mammifère ? NON
Vit il dans l'eau ? OUI
Je pense que c'est une truite.
 C'est bien une truite n'est ce pas ? NON
Alors qu'est ce que c'est ? une écrevisse
Quelle question poseriez vous pour distinguer une truite d'une écrevisse ? Est ce un poisson ?
Quelle serait la réponse à cette question pour une truite ? OUI
Merci ! Je connais maintenant 11 animaux.
...
```

Le programme ci-dessus essaie de deviner, en posant une série de questions, le nom d'un animal que l'utilisateur a choisi. Lorsqu'il échoue, il demande à l'utilisateur le nom de l'animal auquel il avait pensé, et de lui apprendre à le distinguer de celui qu'il avait trouvé.
La base de données de ce programme est un arbre binaire dont les feuilles correspondent aux animaux, et dont chaque nœud interne correspond à une question et possède un fils "OUI" et un fils "NON".
La structure associée à un nœud est la suivante :

```
struct noeud
  { char * text; struct noeud * oui; struct noeud * non; }
```

(Pour les feuilles, les pointeurs **oui** et **non** auront la valeur NULL).

L'arbre initial est construit au lancement du programme à partir d'un fichier texte (voir exemple ci-contre) reproduisant cette structure, sauf que les pointeurs non nuls sont remplacés par des index correspondants au numéros des enregistrements (lignes). La première ligne contient le nombre d'enregistrements, c'est-à-dire le nombre de nœuds de l'arbre. Les lignes suivantes sont les enregistrements numérotés à partir de 1. Pour chaque enregistrement, le premier index correspond à **oui**, le deuxième à **non**. Les feuilles n'ont pas d'index.

```
17
2  11  Est ce un mammifère
3  8   Est ce un ruminant
4  7   Est ce un animal domestique
5  6   Donne t'il de la laine
un mouton
une vache
un buffle
9  10  A t'il des griffes rétractiles
un chat
un chien
12 15  Vit il dans l'eau
13 14  Est ce un poisson
une truite
une écrevisse
16 17  Est-ce un oiseau
un pinson
une abeille
```

1) Ecrire la définition d'un type **ptnod** pointeur vers la structure définie précédemment.

Si on le juge utile, on pourra écrire d'autres fonctions (par exemple pour lire ou écrire un nœud, pour attendre une réponse oui ou non, etc.).
On n'oubliera pas de commenter clairement les listings de toutes les fonctions réalisées, y compris la fonction *main()*, et de faire toute remarque qui s'impose.
Il serait bon de tester chaque fonction dès qu'elle est réalisée ; et pour cela d'écrire une fonction *main()* provisoire (de 2 ou 3 lignes). On ne donnera pas les listings de ces fonctions *main()* provisoires.

2) Ecrire la fonction **lectfich**() de lecture du fichier de données associé à ce programme. Elle aura pour seul argument le nom du fichier. Plus précisément elle devra : ouvrir le fichier, lire le nombre de nœuds, lire les enregistrements correspondant aux différents nœuds, et se terminer en retournant le nombre de nœuds. Elle devra en outre faire toutes les allocations mémoires et tous les tests nécessaires (existence et lisibilité du fichier, cohérence du fichier, mémoire suffisante...) et afficher éventuellement les messages d'erreurs correspondants. Elle devra enfin remplacer l'index lu dans le fichier par le pointeur correspondant.

Indication : Utilisez un tableau dynamique pour stocker cet arbre initial.

3) Ecrire une fonction **posequest**() (elle peut être récursive, mais c'est mieux de la faire non récursive) pour poser une question (ou les questions).

4) Ecrire une fonction **insnouv**() d'insertion d'un nouvel animal dans l'arbre. Cette fonction devra faire tous les tests jugés utiles. Pas de réalloc.

5) Ecrire enfin la fonction **main**() définitive permettant la mise en œuvre du programme complet.

6) (facultatif) Rajouter une fonction **ecrifich**() de réécriture complète du fichier. Elle devra elle aussi faire tous les tests nécessaires, et elle devra être appelée juste avant de quitter le programme. Afin de pouvoir traiter cette question, on pourra rajouter un membre **indx** à la structure. (bonus +3)

7) (facultatif) Modifier la fonction **insnouv**() pour qu'une erreur de mémoire insuffisante ne provoque pas une sortie du programme, mais affiche un message «... je ne pourrai me souvenir de cet animal... mais le jeu continue!...», et faites en sorte que le jeu puisse continuer effectivement. (bonus +2)