ISIMA 1ère année --- TP de C n° 6
Liste chaînée, guide de style et documentation

Première partie : Un programme C qui gère la structure de liste chaînée suivante :

```
typedef struct cellule
{
   char           * ligne;
   struct cellule * suiv;
} cellule_t;
```

1) Ecrire un programme qui lit des chaînes de caractères au clavier, et les insère **en fin** de liste, puis lorsque la saisie est terminée on affiche le résultat. On utilisera de préférence un pointeur sur le début et un pointeur sur la fin de la liste.
   La fonction `fgets` (ou `gets`) vous permet de lire une phrase entrecoupée d'espaces (contrairement au "scanf %s" qui ne lit que des mots).
2) A partir du programme précédant, le modifier pour lire les chaînes de caractères dans un fichier texte plutôt qu'au clavier. On utilisera la fonction `fgets` (qui rajoute le '\n' en fin de chaîne). Tester si votre programme exécutable lit bien le fichier source (texte) de votre programme.

Attention : La combinaison d'un `scanf` et d'un `gets` peut poser problème sur les compilateurs GNU récents. En effet, l'enchaînement des deux instructions suivantes ne fonctionne pas correctement :

```
scanf("%d",&choix);
gets(chaine);
```

Le retour chariot tapé après la saisie d'un int pour le `scanf %d` est interprété comme chaîne de caractères à part entière pour le `gets` qui suit, ce qui implique que cette fonction semble être ignorée. Un `fflush(stdin)` ne solutionne pas sur les compilateurs `gcc` actuels. On peut utiliser la fonction `getchar()` pour lire ce caractère retour chariot et avoir un fonctionnement correct. Une autre solution courante est de procéder comme suit :

```
scanf("%d%*c",&choix);
gets(chaine);
```

La version la plus fiable de `gets` est reste celle donnée par `fgets` lorsque l'on lit sur le fichier d'entrée standard (par défaut le clavier) : `fgets(stdin, NB_MAX_DE_CHARS_A_LIRE, chaine);`

Deuxième partie :

a) Ecrire un programme qui lit sur la ligne de commande le nom d'un fichier texte correspondant à un fichier source en langage C. Vous réutiliserez donc le code de la 1ère partie. Ce nouveau programme produit en sortie un fichier texte avec uniquement les commentaires du fichier source lu.

Vous vous limiterez aux commentaires des entêtes des fonctions, mais vous ajouterez aussi les prototype de chaque fonction (se référer au guide de style ISIMA pour le langage C).

Le résultat doit être aéré et présenté comme une documentation, il pourrait même servir à peu de choses près de fichier « .h » pour le code C qui a été lu. Discuter de l'utilité d'un guide de style, proposez éventuellement une norme par rapport au guide de style distribué en cours de C pour faciliter votre travail d'analyse du fichier source.

Attention : Evitez de tester tous les cas d'erreurs et ne proposez qu'un code simple mais fonctionnel.

b) Ajouter en tête de ce que vous produisez un bloc de commentaire qui précise : le nom du fichier, les auteurs, la date. Ce bloc peut-être : 1) déjà présent sous forme de commentaire dans le code source (c'est le + simple) et dans ce cas ce bloc précise le rôle du programme et non le rôle d'une fonction 2) Pour les DUT généré à partir de balise @author et autres qui auraient été placées dans le code source.

c) Pour ceux qui s'ennuient : Ajouter à la fin des commentaires une statistique « brute » de « qualité du code ». Pour cela on calculera simplement la proportion de commentaires par rapport au code C. Cette proportion sera donnée en nombre de lignes et en nombre de caractères. Réfléchissez à la pertinence de cette mesure.