[LC545]

TP allocations

☞ Dans ce TP, on écrira (par étapes) un seul programme qui exécutera successivement les instructions des questions (1°), (2°) et (3°). La question (4°) modifiera l'ensemble, mais on ne fournira que le dernier listing. On testera le programme à chaque étape, avant de passer à la question suivante. On fera des allocations mémoires là où on le jugera utile, et on prendra toutes les précautions nécessaires pour éviter les plantages que l'utilisateur-testeur-enseignant ne manquera pas de tenter de produire! Le dossier que vous devez rendre ne doit pas être seulement le listing du programme, mais devra être complété par des remarques et explications (et commentaires), le tout constituant des réponses aux questions posées, avec éventuellement un jeu d'essai correspondant à l'exécution du programme tout entier.

On considère un tableau structuré dont la définition est la suivante :

```
#define NBEL 100
struct { char * txt; int nbre; } tabl[NBEL];
```

1) Ecrire quelques instructions permettant d'afficher les tailles d'un entier, d'un pointeur, d'un caractère et enfin du tableau **tabl** précédent. En exécutant ces instructions, vérifier ce qui a été dit en cours.

2) (2.a) Ecrire une boucle[1] qui :

- demande à l'utilisateur de saisir un nombre N, puis un texte (de 250 caractères maximum),
- affiche le nombre N, la longueur[2] L du texte et leur produit NL,
- stocke le nombre N, d'une part, et N fois le texte (concaténé à lui-même), d'autre part, dans un élément du tableau ci-dessus (le parcours de celui-ci sera fait dans l'ordre),
- appelle la fonction **memorymap( )**[3] (avec argument 0) afin d'afficher l'état des allocations mémoires. Reportez vous au manuel (commande **man**) pour avoir des détails sur cette fonction ainsi qu'éventuellement sur celles vues en cours.

Cette boucle se terminera (sans faire les trois derniers points) lorsque l'utilisateur saisira la valeur 0 pour le nombre N.

(2.b) Pour tester cette deuxième partie on choisira[4] bien ses jeux d'essais et on fera au moins une douzaine d'exécutions, chacune correspondant à plusieurs tours de boucles. Vous présenterez sous forme de tableau, dans votre dossier, les valeurs de N, de L, les tailles de mémoire demandées, et les tailles effectivement prises par l'allocation.

(2.c) Expliquer la différence entre les tailles demandées et obtenues, et expliquer comment elle varie et pourquoi. Pour vous aider à comprendre cette différence, vous pouvez faire afficher par votre programme certains octets de la zone allouée.

3) Ecrire une seconde boucle qui parcourt les éléments occupés du tableau, et qui pour chacun de ces éléments :

- affiche le nombre et la longueur du texte,
- libère la mémoire qui avait été allouée,
- affiche de nouveau l'état de la mémoire,
- attend un caractère au clavier pour permettre à l'utilisateur de lire avant de passer à l'élément suivant.

☞ Dans cette question, le parcours devra se faire en utilisant un "pointeur courant" (**pta** pointeur vers un élément du tableau) au lieu d'un indice. Le test de fin de boucle pourra utiliser un autre pointeur **ptend**[5].

4) (facultatif) Remplacer la définition statique du tableau `tabl[NBEL]` par une allocation dynamique qui sera réajustée tous les dix éléments. C'est à dire qu'on alloue d'abord pour dix éléments, puis s'il y a un 11ème élément, on réalloue[6] pour 20 ; puis s'il y en a un 21ème, on réalloue pour 30 ; etc.

---

[1] En choisissant bien les fonctions utilisées afin de minimiser les temps de calcul.
[2] Rappelons que celle-ci est donnée par la fonction **strlen**.
[3] Cette fonction pourra être remplacée sous PC (TurboC) par la fonction **coreleft**() (ou la fonction **farcoreleft**() si on utilise **farmalloc**() ou **farcalloc**()..), bien qu'elle ne donne que la taille de la mémoire restante.
[4] Il est recommandé, entre autres, de prendre N<10000 et un texte de moins d'une ligne.
[5] Prendre garde que sous PC un tel test n'est possible sans risque que si on utilise des pointeurs "géants" (**huge pointers**).
[6] Rappelons qu'une réallocation qui échoue, conserve les données dans la précédente zone. Pensez à sauvegarder le pointeur avant de réallouer.