

TP12

Langage C (LC4)

Semaine du 22 avril 2013

Répétition des listes

Dans ce TP, on travaille avec le type de listes suivant :

```
typedef struct Liste{
    int valeur;
    struct Liste *suivant;
} *list;
```

Par convention, la liste vide est représentée par le pointeur NULL.

Question 1. Écrire une fonction `void affiche_list(list l)` qui affiche les éléments d'une liste `l` sur le terminal.

Question 2. Écrire une fonction `list lire_list()` qui lit une liste d'entiers du terminal un par un, et en crée une liste qui est renvoyée. La lecture arrête quand l'utilisateur entre une ligne vide.

Question 3. Écrire une fonction `void libere_list(list l)` qui libère la mémoire occupée par une liste `l`.

Pointeurs vers des fonctions

Question 4. Quel est le type d'un pointeur vers une fonction prenant un `int` en argument et renvoyant un `int` en résultat ?

Question 5. Écrire une fonction `int forall(list l, int (*predicate)(int))` qui prend en argument un pointeur `l` vers une liste, et un pointeur `predicate` vers une fonction, et qui détermine si la fonction appliquée à tous les éléments de la liste retourne une valeur non-zéro.

Question 6. Écrire une fonction `int exists(list l, int (*predicate)(int))` qui détermine si il existe un élément `n` dans `l` tel que `predicate(n)` est différent de zéro.

Question 7. Écrire une fonction `list filter(list l, int (*predicate)(int))` qui crée une liste contenant tous les éléments `n` de `l` tel que `predicate(n)` est différent de zéro, et renvoie cette liste.

Attention : la liste originale ne doit pas être détruite.

Question 8. En utilisant la question précédente, écrire une fonction `void affiche_filter(list l)` qui affiche la sous-liste des éléments impairs de `l` et la sous-liste des éléments positifs de `l`. Écrivez les fonctions auxiliaires nécessaires.

Veillez à libérer tout mémoire allouée dynamiquement qui ne sera plus utilisé.

Question 9. Écrire une fonction `void split(list l, int (*predicate)(int), list *r, list *s)` qui crée deux listes – une contenant tous les éléments `n` de `l` tel que `predicate(n)` est différent de zéro, et l'autre contenant les éléments tel que `predicate(n)` est égal à zéro. Les deux listes doivent être renvoyées par référence en utilisant les paramètres `r` et `s`. La liste originale ne doit pas être détruite.

Question 10. En utilisant la question précédente, écrire une fonction `void affiche_split(list l)` qui affiche la sous-liste des éléments impairs de `l` et la sous-liste des éléments pairs de `l`.

Veillez à libérer tout mémoire allouée dynamiquement qui ne sera plus utilisé.