Algorithmique - ITII CNAM Alsace 1A
Chaînage : piles, files et listes

On considère la structure de données suivante en C :

```
typedef struct liste{int x;struct liste *suivant;} Sliste, *Liste;
```

Fonctions de base sur les listes

Implanter les opérations de base sur les listes :

1. construire la liste vide,
2. ajouter un élément en tête de la liste,
3. retourner la tête d'une liste,
4. tester si une liste est vide,
5. supprimer un élément donné en argument,
6. rechercher un élément dans la liste,
7. calculer la longueur (nombre d'éléments de la liste) de manière récursive.
8. calculer de nouveau la longueur, de manière itérative avec une boucle `while`.

Fonctions avancées

1. Ecrire une fonction qui concatène 2 listes l1 et l2 : `Sliste *append(Sliste *l1, Sliste *l2);`.
2. Ecrire une fonction qui retourne le i^{eme} élément d'une liste : `int ieme(Sliste *l, int i);`.

Parcours de listes

1. Ecrire une fonction qui calcule (récursivement) la somme des éléments d'une liste donnée en argument `int somme(Sliste *l);`.
2. Ecrire une fonction qui multiplie par 2 tous les éléments d'une liste : `void mult2(Sliste *l);`.
3. Ecrire une fonction qui prend 2 listes en arguments et ajoute 2 à 2 les éléments. Par exemple, avec les listes "2,4,5" et "1,0,1", on doit obtenir "3,4,6". Si les listes ne sont pas de même longueur, on complètera la liste plus courte avec des zéros à la fin : `Sliste *couplage(Sliste *l1, Sliste *l2);`
4. Ecrire une fonction qui retourne une liste avec les éléments dans l'ordre inverse, "2,3,4" devient "4,3,2" : `Sliste *renverse(Sliste *l);`.

Listes triées

On va ajouter une contrainte dans nos listes. On suppose qu'elles sont triées. L'insertion ne se fait donc plus en tête, mais à une position à déterminer pour que la liste reste triée. Par exemple, dans le chaînage "2,7,9,12" , l'insertion de 8 se fait entre le maillon portant l'entier 7 et le maillon portant l'entier 9.

1. Programmer une nouvelle fonction d'insertion dans une liste, qui permette de conserver l'invariant que la liste est triée.

2. Modifier les opérations de recherche et de suppression pour qu'elles s'arrêtent dès que l'on sait que la recherche ou la suppresion échouera.

3. Programmer une fonction de fusion de 2 listes triées l1 et l2 en une nouvelle liste triée : `Sliste *fusion (Sliste *l1, Sliste *l2);`.

4. A l'aide des fonctions d'insertion et de fusion, écrire une fonction de tri d'une liste : `Sliste * tri(Sliste *l);`.

Liste circulaire et Flavius Josëphe / Josephus

Implantation des listes circulaires :
La manière la plus simple d'implanter des listes circulaires est de reprendre le type de données des listes, de construire la liste de manière non circulaire et de faire pointer le pointeur positionné à `NULL` qui indique la fin de la liste vers la tête de la liste.

Petite histoire :
Josephus Flavius était un historien romain du premier siècle. Durant une guerre lui et 40 de ses camarades soldats furent pris au piège, entouré par les troupes ennemies. La légende raconte que le groupe encerclé préféra se suicider plutt que d'tre capturé. Ainsi Josephus et ses soldats formèrent un cercle et décidèrent de se tuer mutuellement et successivement, de manière ce qu'une personne tue la troisième personne sur sa gauche, que la personne droite du mort tue son tour la troisième personne sur sa gauche, ainsi de suite jusqu' ce qu'il ne reste qu'un seul survivant. Restant seul, ce dernier est censé se suicider lui-même. Josephus, qui ne souhaitait pas mourir, trouva rapidement la place sûre, c'est-à-dire la place de la dernière personne debout, sans que quiconque ne reste pour le tuer. Ainsi il resta en vie et put par la suite raconter cette légende. Trouver cette place sûre est maintenant appelé le problème de Josephus.

Nous allons généraliser le problème pour un cercle composé de N soldats, il faut alors trouver où va se placer Josephus. Programmer une solution qui, étant donné un générateur de liste de taille N et une fonction `flavius`, paramétré par une taille N, renvoie la position où doit se mettre notre cher historien.

La liste circulaire contient maintenant des entiers correspondant à la position dans la liste.

Files

L'implantation des files se fait avec des listes. Il faut cependant conserver 2 pointeurs, un vers la tête et l'autre vers la queue de la file.

1. Proposer un structure de données à ajouter par dessus les listes pour avoir des files.

2. Ecrire les opérations d'ajout (d'un côté de la file) et de suppression (de l'autre côté de la file).

3. Ecrire une fonction pour tester si la file est vide.

4. Calculer la taille de la file (par récursivité).

5. Faire une variante itérative du calcul de la taille de la file.