Programme d'informatique n° 2

Recherche dans un liste chaînée

Le programme ci-dessous (disponible : `~bguil/public/prog2.c`) fait une suite de recherche dans une liste chaînée. Il compte également le coût global de cette série de recherche en comptant le nombre de comparaisons de l'élément recherché avec les éléments de la liste.

```
#include<stdio.h>
#define N 200 /* nombre max de recherche sur la liste */

typedef enum {
  false,
  true
} boolean;

struct liste_chainee
{ int valeur;
  struct liste_chainee *suivant;
} ;
typedef struct liste_chainee ELEMENT;
typedef ELEMENT *LISTE;

/* renvoie true ssi p est la liste vide. */
boolean est_vide (LISTE p)
{ return (p == NULL); }

/* creation de la liste chainee 1 --> 2 --> 3 .... --> k (on suppose k > 0) */
LISTE list_1_k (int k)
{
  int i=k;
  LISTE tmp, l= NULL;
  while (i > 0) {
    tmp=l;
    l = (LISTE) malloc (sizeof (ELEMENT));
    l->valeur=i; l->suivant=tmp; i--;
  }
  return l;
}

/* affiche sur une ligne les elements de la liste */
void affiche_liste (LISTE l)
{
  while (!est_vide(l)) {
    printf ("%d  ", l->valeur);
    l = l->suivant;
  }
}
```

```
/* retourne true si l'element x est dans la liste et met sa position dans *position_ptr */
/* incremente *cpt du nombre de comparaisons effectuees */
boolean recherche (int x, LISTE l, int *position_ptr, int *cpt)
{
  int succes = false, fini = false, pos = 1;
  while (!fini) {
    if (est_vide(l))
      fini=true;
    else {
      (*cpt)++;
      if (l->valeur == x) {
        succes = true; fini = true; *position_ptr = pos;
      }
      else {
        pos++; l = l->suivant;
  } } }
  return (succes);
}

main ()
{
  int tab[N],trouve,i,longueur,nbre_rech,pos,cpt;
  LISTE l;
  scanf ("%d", &longueur);    /* longueur de la liste initiale */
  scanf ("%d", &nbre_rech);   /* nombre de recherche sur la liste */
  for (i=0; i<nbre_rech; i++)
    scanf ("%d", &tab[i]);
  cpt = 0; l = list_1_k (longueur);   /* initialise cpt et l */
  for (i=0; i<nbre_rech; i++)
    trouve = recherche (tab[i],l,&pos, &cpt);
  affiche_liste (l); printf (" (%d)\n", cpt);
  l = list_1_k (longueur);
}
```

Lorsque certains éléments sont recherchés plus souvent que d'autres, il y a de meilleures façons de s'y prendre ; c'est le principe de la recherche "auto-adaptative" qui modifie la liste afin que les éléments recherchés le plus souvent soient proches du début de la liste, d'où une complexité plus faible :

1. après chaque recherche réussie, on place l'élément qui a été recherché en tête de la liste ;
2. après chaque recherche réussie, l'élément recherché progresse d'une place vers le début de la liste (s'il n'est pas déjà en tête).

Nous vous demandons d'ajouter deux fonctions de recherche mettant en œuvre ces méthodes. Votre programme prendra la même entrée que le modèle proposé et il répondra dans l'ordre la liste obtenue à l'issue des recherches, suivie du nombre de comparaisons effectuées, pour chacun des types des recherches (simple, mise en tête et décalage d'un rang).

Jeu de tests.

Sur les données ci-dessous, le programme du corrigé donne les réponses suivantes :

Entrée	Sortie
10 4 8 8 8 8	1 2 3 4 5 6 7 8 9 10 (32) 8 1 2 3 4 5 6 7 9 10 (11) 1 2 3 8 4 5 6 7 9 10 (26)