Contrôle 1, durée 1h30

Lundi 20 Octobre 2014

Exercice 1. Affichages (6 pt)

```
struct toto{
  int t[3][7];
  int* p;
};

void modif(struct toto S, int T[3][7]){

  /*Affichage 2*/
  printf("%d %d %d %d\n", sizeof(S), sizeof(T), sizeof(S.t), sizeof(S.p));

  S.t[0][0] = 1;
  S.p[0] = 1;
  T[0][0] = 1;
}

int main(void){
  struct toto S;
  int T[3][7]={{0}};
  S.t[0][0] = 0;
  S.p = malloc(sizeof(int));
  S.p[0] = 0;

  /*Affichage 1*/
  printf("%d %d %d\n", T[0][0], S.t[0][0], S.p[0]);

  modif(S, T);

  /*Affichage 3*/
  printf("%d %d %d\n", T[0][0], S.t[0][0], S.p[0]);

  free(S.p);
  return 0;
}
```

a. (5 pt) Qu'affichent dans l'ordre les 3 appels à la fonction `printf` du programme précédent ? Justifier votre réponse pour les affichages 2 et 3. Pour l'usage de `sizeof`, on suppose que les entiers et les pointeurs sont codés sur 4 octets.

b. (1 pt)
Qu'affiche l'instruction suivante ? Aucune justification n'est demandée.

```
char* s = "toto"
printf(&0["s%ss\n"], s+3);
```

Exercice 2. Opérations sur les listes chaînées (14 pt)

On donne la strucure de liste chaînée stockant des entiers

```
struct element{
  int val;
  struct element *suiv;
};
typedef struct element *Liste;
```

Écrire les 4 fonctions suivantes de manière *récursive* (0.5 point par question sera déduit sinon) :

a. (2 pt)

```
int somme(Liste l);
```

qui retourne la somme des valeurs contenues dans la liste.

b. (1.5 pt)

```
int occurence(Liste l, int e);
```

qui compte le nombre d'occurences de `e` dans la liste, c'est-à-dire le nombre de fois où la valeur `e` est présente dans la liste.

c. (2 pt)

```
int valMax(Liste l)
```

qui renvoie la valeur maximale contenue dans la liste, et 0 si la liste est vide.

d. (2 pt)

```
Liste supprimer(Liste l, int e);
```

qui supprime *toutes* les valeurs `e` dans la liste. On suppose que la liste est dans la mémoire dynamique, il faut donc libérer cette mémoire quand on supprime les éléments.

Dans une liste chaînée, le type des valeurs stockées est quelconque. On peut notamment stocker un ensemble de listes dans une liste, ce qui constitue une *liste de listes*.

e. (1 pt) Dessiner à l'aide de flêches et de rectangle à 2 champs une réprésentation d'une liste qui contient la liste d'entiers $\{3, 9, 7\}$ et la liste d'entiers $\{8, 5, 6, 2\}$.

f. (1 pt) Définir une structure `LListe` qui permet de coder une liste qui contient des listes `Liste`.

g. (0.5 pt) Quelle valeur contient une liste de listes vide ?

h. (2 pt) Écrire la fonction

```
LListe ajoutDebut(LListe ll, Liste l)
```

qui ajoute la liste d'entiers `l` à la liste de listes `ll` dans la mémoire dynamique.

i. (2 pt) Écrire la fonction

```
int nbElement(LListe ll)
```

qui retourne le nombre d'entiers contenus dans la `LListe` notée `ll` de manière récursive sans faire appel à une autre fonction qu'à elle-même.