TP3 - Allocation de mémoire

Exercice 1: Supposons la structure déclarée comme suit :

```
struct liste{
  int val;
  struct liste *ptr;
}
```

1. Écrire une fonction qui prend en paramètre un pointeur vers une `struct liste`, considéré comme la tête d'une liste chaînée d'entiers, et affiche tous les éléments de la liste.
2. Écrire une fonction qui prend en paramètre le même type de pointeur, plus un entier `n`, crée un nouveau noeud avec `val = n` et *insère* le nouveau noeud à la 5e position dans la liste (donc entre le 4e noeud et le 5e).
3. Écrire une fonction qui prend en paramètre le même type de pointeur, plus deux entiers (appelons-les `n` et `pos`), et insère l'entier `n` entre le `(pos-1)`-ième et le `pos`-ième noeud de la liste.
4. Écrire un programme qui lit une suite d'entiers saisie au clavier et crée une liste chaînée (dans l'ordre de saisie), puis lit encode deux entiers `n` et `pos` et insère `n` à la position `pos` en appelant la fonction écrite au point précédent.
5. Écrire une fonction qui prend en paramètre le même type pointeur plus un entier `n` et *supprime* l'entier qui se situe à la position `n` dans la liste chaînée. *Attention !* Cette fonction devrait fonctionner correctement même si `n` est plus grand que le nombre de noeuds dans la liste ! (et donc ne rien faire dans ce cas-là !)

   Pour la construction de la liste chaînée pendant la lecture de la suite d'entiers au clavier, reprendre le code vu en cours pour la construction d'une liste chaînée dans ordre de saisie.

Exercice 2:

1. Déclarez un pointeur d'entiers. Puis allouez-lui de la mémoire avec `malloc`, et affichez son adresse. Désallouez-le, et réallouez-le immédiatement. Est-ce qu'il est alloué à la même adresse ?
2. Refaites le point précédent, mais cette fois-ci en allouant plusieurs pointeurs de tailles différentes, puis en les désallouant et réallouant à nouveau. Que constate-t-on ?
3. Déterminez la taille de la zone de gestion qui se trouve à la fin de chaque cellule réservée par un appel `malloc`. Pour cela, appeler deux fois de suite `malloc` et afficher la différence entre les deux pointeurs retournés, cela en passant différentes tailles de données en paramètre à `malloc`.
4. *(Facultatif)* Donnez l'erreur (de compilation ou exécution) qui est issue par le code suivant (**lorsqu'il est correctement placé dans un main !**) :

   ```
   char *c;
   c="abcd";
   c[2]=0;
   ```

   Donnez une explication pour ce comportement.

Exercice 3: Écrire une fonction qui lit une suite de chaînes de caractères (terminée par la chaîne composée du chiffre 0), et crée une liste chaînée qui contient les chaînes de caractères dans l'ordre inverse de saisie. Pour cela, adapter le code vu en cours pour la construction d'une liste chaînée en ordre inverse de saisie.
