TD3 - Allocation de mémoire

Exercice 1:

1. Écrire une fonction *clone* qui crée une “clone” de la chaîne de caractères passée en paramètre (un seul paramètre). La clone est créée par allocation de mémoire à l’intérieur de la fonction.
2. Écrire un programme qui lit une chaîne de caractères (avec `scanf`), crée une clone de cette chaîne en appelant la fonction écrite au point précédent et affiche cette clone.

Exercice 2: 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 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 !)
5. 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.

   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 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.

Exercice 4:

1. Écrire une fonction *concat* qui reçoit en paramètre deux chaînes de caractères et **crée** (en allouant de la mémoire) une troisième chaîne dans laquelle la fonction copie les deux chaînes.
2. Écrire un programme qui lit deux chaînes de caractères (avec `scanf`), les concatène en appelant la fonction écrite au point précédent et affiche le résultat.
