A.2 Liste et chaînage

1. En vous inspirant de l'implémentation d'une liste par curseur vue en cours, proposez une "vraie" implémentation chaînée de la liste, qui utilise directement l'allocation dynamique de mémoire (`malloc()` et `free()` en C). Pour des raisons de lisibilité, on conservera les fonctions `newLibre()` et `freeLibre()`, même si leur code est substantiellement réduit. On choisira de stocker la longueur en permanence, pour réduire le coût et faciliter les contrôles. On aura donc un type liste défini, par exemple, ainsi (ici `cliste` pour une liste de caractères) :

```
struct s_cellule {
  char val ;
  struct s_cellule * sv ;
} ;
typedef struct s_cellule cellule ;
#define nil (cellule *) 0

struct s_cliste {
  cellule * tete ;
  int l ;
} ;
typedef struct s_cliste cliste ;
```

2. En utilisant les primitives de manipulation de liste définies, proposer une fonction qui affiche le contenu de la liste en commençant par le dernier élément. Complexité de l'opération (en cas d'implémentation contigüe, en cas d'implémentation chaînée) ?
3. Pour réduire certains coûts dans une implémentation chaînée pour une liste, on envisage, lors des suppressions, d"'effacer" la valeur stockée (par exemple en la remplaçant par une valeur absente de la liste) sans faire les décalages. Quelles sont les conséquences sur les autres primitives ?