## A.1 Liste

1. On suppose les fonctions suivantes définies pour la manipulation d'une liste de réels (`rliste`) (`l` est une rliste, `p` est une position, `e` est un réel) : `longueur(l)`
   `inserer(l,e,p)`
   `supprimer(l,p)`
   `element(l,p)`

   (a) Ecrire une fonction qui prend une rliste en paramètre, et trie la liste. Deux implémentations seront réalisées : l'une qui trie la liste "sur place", l'autre qui crée une nouvelle liste triée. Comparer les coûts dans les deux cas.

   (b) Ecrire une fonction qui supprime tous les doublets de la liste (sans la réordonner).

2. Pour résoudre le problème de la borne statique du tableau, ajouter à la fonction d'insertion un contrôle sur la taille, qui réalloue un tableau plus grand lorsque la taille maximale est atteinte.

3. "Padding". Modifier la fonction d'insertion de telle sorte que si on demande d'insérer un élément à une position $p$ plus grande que la taille actuelle, la liste est complétée jusqu'à la position $p$ avec une valeur "neutre", par exemple 0.

4. Pour trier une liste, on a souvent besoin d'une fonction d'échange qui intervertit les valeurs de deux positions dans la liste. Proposer deux versions de fonction d'échange : l'une écrite au moyen des primitives déjà écrites, l'autre écrite directement en accédant à la structure de données. Discuter la complexité.

rliste.h

```
#define RLISTE_MAX 100
struct s_rliste {
  float tab[RLISTE_MAX] ;
  int l ;
} ;
typedef struct s_rliste rliste ;
```

rliste.c

```
#include "rliste.h"

int longueur (rliste l)
{
  return l.l ;
}

float element(rliste l, int p)
{
  // Hypothèse : p in [1,longueur(l)]
  return l.tab[p-1] ;
}
```

```
void inserer(rliste *l, int p, float v)
{
  // Hypothèses : p in [1,longueur(l)+1]
  //              longueur(l) < RLISTE_MAX
  int i ;
  for (i=l->l-1 ; i>=p-1 ; i--)
    l->tab[i+1] = l->tab[i] ;
  l->tab[p-1] = v ;
  l->l++ ;
}

void supprimer(rliste *l, int p)
{
  // Hypothèse : p in [1,longueur(l)]
  int i ;
  for (i=p-1 ; i < l->l-1 ; i++)
    l->tab[i] = l->tab[i+1] ;
  l->l-- ;
}
```