1. **Complexité**

   (a) Ce programme supprime toutes les occurrences de "a" et "e" dans le tableau, en commençant par la fin, et en décalant le contenu.

   (b) Sa complexité est de l'ordre de max^2 (deux boucles imbriquées, chacune bornée par la longueur du tableau — max).

   (c) Au moins trois pistes possibles (pour avoir un algorithme d'ordre max) :
   - On crée un second tableau dans lequel on copie le premier, sans recopier "a" et "e" :
     ```
     for (i=1, j=1 ; i<max ; i++)
       if (t[i] != 'e' && t[i] != 'a') t[j++] = t[i] ;
     ```
   - On transforme le tableau en liste chaînée (par curseur ou par pointeur) de façon à avoir des suppressions en temps constant.
   - Un peu plus « malin » (mais pas forcément plus lisible et maintenable) : on part de la gauche, et on mémorise le nombre de cases dont il faut décaler au fur et à mesure.
     ```
     int offset = 0 ; // décalage
     for (i=1 ; i<max ; i++) {
       while (t[i+offset]=='e' || t[i+offset] == 'a') offset++ ;
       t[i] = t[i+offset] ;
     }
     ```

2. **Représentation d'ensembles d'entiers sous forme de tableau de booléens**
   N.B. : On dira qu'un entier $i \in [1, \texttt{MAX}]$ appartient à un ensemble donné $\texttt{e}_1$ (c'est-à-dire un tableau particulier de booléens) si et seulement si $\texttt{e}_1[i]$ vaut true.

   (a) **Appartenance**
   ```
   appartient := ens[elt] ;
   ```

   (b) **Initialisation**
   ```
   for i:=1 to MAX do
     ens[i] := false ;
   ```

   (c) **Cardinal**
   ```
   c := 0 ;
   for i := 1 to MAX do
     if ens[i] = true then c := c + 1 ;
   cardinal := c ;
   ```

   (d) **Union**
   ```
   initialisation(e3) ;
   for i := 1 to MAX do
     if e1[i] = true or e2[i] = true then e3[i] = true ;
   ```

   (e) **Intersection**
   ```
   initialisation(e3) ;
   for i := 1 to MAX do
     if e1[i] = true and e2[i] = true then e3[i] = true ;
   ```

3. *Cf. cours.*