```c
#include<stdio.h>

// case n° 0 non utilisée
int t1[] = {-1, 7, 17, 1, 2, 5, 10, 6, 8, 12, 11, 4, 9, 13, 3, 18} ;
int n1 = 15 ;
int t2[] = {-1, 7, 12, 44, 91, 23, 31, 10, 67} ;
int n2 = 8 ;
int t3[] = {-1, 8, 6, 9, 12, 8, 4, 9, 7, 5, 14, 1} ;
int n3 = 11 ;

int *tabs[] = {t1, t2, t3} ;
int *ns[] = {&n1, &n2, &n3} ;
int nbs = 2 ;


void aff_tab(int t[], int d, int f) {
  int i ;
  for (i=d ; i<=f ; i++)
    printf("% 4d", t[i]) ;
  printf("\n") ;
}

void echange_tab(int t[], int i, int j) {
  int temp = t[i] ;
  t[i] = t[j] ;
  t[j] = temp ;
}

int posPivot(int t[], int d, int f)
// deux objectifs: recherche un pivot
// (la plus grande des 2 premières valeurs différentes)
// et renvoie -1 si le tableau ne contient que des valeurs identiques
// (il est alors déjà trié)
{
  int v1 = t[d] ;
  int k ;
  for (k = d+1 ; k <= f ; k++)
    if (t[k] < v1)
      return d ;
    else if (t[k] > v1)
      return k ;
  // à ce point, on n'a pas trouvé de valeurs différentes
  return -1 ;
}
```

```c
int partition(int t[], int d, int f, int pivot)
// Avec deux indices qui délimitent les extrémité déjà bien
// disposée par rapport au pivot, on échange les valeurs mal
// placées jusquà ce que les indices se rejoignent
{
  int gauche = d ;
  int droite = f ;
  do {
    echange_tab(t, gauche, droite) ;
    while (t[gauche] < pivot)
      gauche++ ;
    while (t[droite] >= pivot)
      droite-- ;
  } while (gauche <= droite) ;
  return gauche ;
}


void quicksort(int t[], int d, int f)
// le tableau reçu en paramètre est réorganisé par
// rapport à un pivot, et le pgm est relancé sur les
// deux parties.
// condition d'arrêt: pas de pivot (= une seule valeur dans le tableau)
{
  int k, m ;

  k = posPivot(t,d,f) ;
  if (k != -1) {
    m = partition(t,d,f,t[k]) ;
    quicksort(t, d, m-1) ;
    quicksort(t, m, f) ;
  }
}


int main()
{
  int i;

  for (i=0 ; i<=nbs ; i++) {
    aff_tab(tabs[i], 1, *ns[i]) ;
    quicksort(tabs[i], 1, *ns[i]) ;
    aff_tab(tabs[i], 1, *ns[i]) ;
  }
}
```