```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} ;



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 plus_petit_fils(int t[], int d, int f)
{
  int fg=2*d ;
  int fd=2*d+1 ;
  if (fg > f)
    return -1 ;
  if (fd > f)
    return fg ;
  if (t[fg] <= t[fd])
    return fg ;
  else
    return fd ;
}

void glissade(int t[], int d, int f)
// hypothèse: la partie [d+1,f] du tableau est un tas
// pour placer l'élément t[d] correctement, on l'échange
// avec le plus petit de ses deux fils, et on recommence.
// Version récursive
{
  int x = plus_petit_fils(t, d, f) ; // renvoie -1 si pas de fils
  if (x == -1) return ;
  if (t[d] < t[x]) return ;
  echange_tab(t, d, x) ;
  glissade(t, x, f) ;
}
```

```c
void fabrique_tas(int t[], int d, int f)
// soit m = n div 2
// les m dernières cases d'un tableau forment un tas par hypothèse
// on peut donc utiliser glissade() sur le presque-tas t[m-1, n]
// alors t[m-1, n] devient un tas, et on peut appliquer glissade()
// à t[m-2, n].
{
  int m = d+f / 2 ;
  int i ;
  for (i = m-1 ; i>=1 ; i--)
    glissade(t, i, f) ;
}


void tri_dans_tas(int t[], int d, int f)
// on part d'un tas [1,n], on échange la racine (ie le minimum)
// avec t[n], et on applique la glissade sur [1,n-1].
// on recommence, en échangeant la racine avec t[n-1].
{
  int i ;
  for (i=f ; i>1 ; i--) {
    echange_tab(t, 1, i) ;
    glissade(t, 1, i-1) ;
  }
}

void heapsort(int t[], int d, int f)
{
  fabrique_tas(t, d, f) ;
  tri_dans_tas(t, d, f) ;
}


int main()
{
  int i;

  for (i=0 ; i<=2 ; i++) {
    aff_tab(tabs[i], 1, *ns[i]) ;
    heapsort(tabs[i], 1, *ns[i]) ;
    aff_tab(tabs[i], 1, *ns[i]) ;
  }
}
```