```
1
2 ////////////////////////////////////////////////////////
3 // Codes from Algorithms in C Parts 1-4 by R. Sedgewick //
4 ////////////////////////////////////////////////////////
5 typedef int Item;
6 #define key(A) (A)
7 #define less(A, B) (key(A) < key(B))
8 #define exch(A, B) { Item t = A; A = B; B = t; }
9 #define compexch(A, B) if (less(B, A)) exch(A, B)
10
11 int partition(Item a[], int l, int r) {
12     int i = l - 1, j = r; Item v = a[r];
13     for (;;) {
14         while (less(a[++i], v));
15         while (less(v, a[--j])) if (j == l) break;
16         if (i >= j) break;
17         exch(a[i], a[j]);
18     }
19     exch(a[i], a[r]);
20     return i;
21 }
22
23 void quicksort(Item a[], int l, int r) {
24     int i;
25     if (r <= l) return;
26     i = partition(a, l, r);
27     quicksort(a, l, i - 1);
28     quicksort(a, i + 1, r);
29 }
30 ////////////////////////////////////////////////////////
31 #define push2(A, B)  push(B); push(A);
32 void quicksort(Item a[], int l, int r) {
33     int i;
34     stackinit(); push2(l, r);
35     while (!stackempty())   {
36         l = pop(); r = pop();
37         if (r <= l) continue;
38         i = partition(a, l, r);
39         if (i - l > r - i) {
40             push2(l, i - 1); push2(i + 1, r);
41         }
42         else {
43             push2(i + 1, r); push2(l, i - 1);
44         }
45     }
46 }
47 ////////////////////////////////////////////////////////
48 #define M 10
49 void quicksort(Item a[], int l, int r) {
50     int i;
51     if (r - l <= M) return;
52     exch(a[(l + r) / 2], a[r - 1]);
53     compexch(a[l], a[r - 1]);
54     compexch(a[l], a[r]);
55     compexch(a[r - 1], a[r]);
56     i = partition(a, l + 1, r - 1);
57     quicksort(a, l, i - 1);
58     quicksort(a, i + 1, r);
59 ////////////////////////////////////////////////////////
60 void sort(Item a[], int l, int r) {
61     quicksort(a, l, r);
62     insertion(a, l, r);
63 }
64
```