```
1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 5 #define CAP 1000000000
 7 void input (long int*, long int nr_elements);
 8 void sort (long int*, long int nr_elements);
 9 void output (long int*, long int nr_elements);
10 void *xcalloc (long int, long int);
11
12 int main (int argc, char *argv[])
13 {
14
        long int nr elements;
15
        long int *arr;
16
       if (argc != 2)
17
18
            fprintf (stderr, "Usage error:%s nr_elements\n",
19
20
                 argv[0]);
21
            exit (EXIT_FAILURE);
        }
22
23
24
       nr_elements = atol (argv[1]);
25
        if (nr_elements <= 0)</pre>
26
27
            fprintf (stderr, "array dimension must be positive\n");
28
            exit (EXIT_FAILURE);
29
        }
30
31
        arr = (long int*) xcalloc (nr_elements, sizeof (long int));
32
        input (arr, nr_elements);
33
        sort (arr, nr_elements);
34
        output (arr, nr_elements);
35
36
        exit (EXIT_SUCCESS);
37 }
38
39 void input (long int *arr, long int nr_elements)
40 {
41
        long int cnt;
42
43
        srand (time (0));
44
        for (cnt=0; cnt < nr_elements; cnt++)</pre>
45
46
            arr[cnt] = rand () % CAP;
47
        }
48 }
50 void output (long int *arr, long int nr_elements)
51 {
52
        long int cnt;
```

```
\underline{\dots} oot\usr\src\ds\_alg\batch\_codes\C\3\_quick\_sort\sort\_main.c
```

```
2
```

```
53
54
       for (cnt=0; cnt < nr_elements; cnt++)</pre>
55
56
            printf("arr[%ld]:%ld\n", cnt, arr[cnt]);
57
        }
58 }
59
60 void *xcalloc (long int nr_elements, long int size_per_element)
61 {
62
       void *tmp;
63
       tmp = calloc (nr_elements, size_per_element);
64
65
       if (!tmp)
66
67
           fprintf (stderr, "xcalloc:fatal:out of memory\n");
68
            exit (EXIT_FAILURE);
69
70
        }
71
72
       return (tmp);
73 }
74
```