CS29003 ALGORITHMS LABORATORY
Assignment No: 6
Last Date of Submission: 02–September–2015

In this exercise, we deal with an expected linear-time sorting algorithm for integers (the algorithm can be easily adapted to floating-point numbers). Let A be an array of n integers, that we want to sort. We assume that the elements of A are uniformly randomly chosen from some interval $[c, d]$. The endpoints c and d are not known to us. We compute the minimum m and the maximum M in the array A. Every element a in A is guaranteed to lie in the subinterval $[m, M]$ of $[c, d]$. Moreover, for any x in $[m, M]$, an array element a equals x with (conditional) probability $1 / (M - m + 1)$. Therefore, we may proceed with impunity after making the assumption that the elements of A are uniformly randomly chosen from the easily computable interval $[m, M]$. If $M - m$ is $O(n)$, we can use counting sort that will run in linear time. So assume that $M - m \gg n$ (in fact, $M - m = \omega(n)$).

We may break the interval $[m, M]$ into n (almost) equal-sized subintervals. The uniform distribution of the elements of A will imply that we *expect* exactly one element of A to belong to each subinterval. In practice, however, some subintervals may be empty (not populated by elements of A), whereas some subintervals may contain multiple elements of A. To cope with this, the elements of A belonging to a subinterval are organized as a linked list. We use an array L of list headers, one for each subinterval. Each sublist is expected to contain only O(1) array elements. We sort each sublist in O(1) expected time (using any sorting algorithm), and concatenate the sublists to obtain the final sorted output.

Now that we are ready to handle linked lists, we make a final modification. Let us choose a small constant integer k (take $k = 10$ for this assignment). Let us have sublists of expected size k (instead of one as in the last paragraph). The number of sublists (subintervals) will then be

$$l = \text{ceiling}(n / k) .$$

Finally, the size (length) of each subinterval will be

$$s = \text{ceiling}((M - m + 1) / l) .$$

That is, we have the subintervals $[m, m+s)$, $[m+s, m+2s)$, $[m+2s, m+3s)$, and so on. An element a in A will go to the sublist for the interval $[m+is, m+(i+1)s)$, where

$$i = \text{floor}((a - m) / s) .$$

Part A: Write a function *sortlist*() to sort a linked list of integers. You can use any sorting algorithm (even a non-optimal one).

Part B: Divide A into l sublists following the algorithm mentioned above. To do this, write a function *subdivide*() that first computes m, M, l, and s, then creates l sublist headers, and finally inserts elements of A to appropriate sublists. Insertion should be at the beginning of the lists (so each insertion can finish in O(1) time).

Part C: Sort each sublist using the function of Part A, and concatenate the sorted sublists back in A. Write a function *sortandwrite*() for this purpose.

Part D: Write a *main*() function to do the following:

- Read n and the elements of the array A from the user. Print A.
- Call the function of Part B in order to divide A into sublists. Print the sublists.
- Call the function of Part C to store the final output in A. Print A.

Submit a single C/C++ source file solving all the parts. Do not use any global or static variable or array.

Sample Output

```
n = 75

+++ Reading elements of A
    7979  8594  5128  3552  6198  4972  9657  5627  3752  4575
    7203  2411  4267  7585  1309  2005  4393  9107  8869  1977
    7360  7314  1474  4594  5694  7830  8398  6356  9858  9107
    4474  7837  5053  8602  7741  1251  9927  4750  3231  3679
    5678  9434  2442  8945  7019  2751  7302  7764  8210  6172
    8741  2923  9838  6568  6517  5532  4398  4915  8240  4257
    1374  9067  8446  5428  5021  3540  3031  4948  7290  5262
    4979  2968  2048  3773  1913

+++ Dividing the array into sublists
    Sublist( 0): 1913  2048  1374  1251  1474  1977  2005  1309
    Sublist( 1): 2968  3031  2923  2751  2442  3231  2411
    Sublist( 2): 3773  3540  4257  4398  3679  4474  4393  4267  3752  3552
    Sublist( 3): 4979  5262  4948  5021  5428  4915  5532  4750  5053  4594  4575  4972  5128
    Sublist( 4): 6517  6568  6172  5678  6356  5694  5627  6198
    Sublist( 5): 7290  7302  7019  7741  7314  7360  7585  7203
    Sublist( 6): 8446  8240  8741  8210  7764  8602  7837  8398  7830  8594  7979
    Sublist( 7): 9067  9838  8945  9434  9927  9107  9858  8869  9107  9657

+++ Sorting the sublists
    Sublist( 0): 1251  1309  1374  1474  1913  1977  2005  2048
    Sublist( 1): 2411  2442  2751  2923  2968  3031  3231
    Sublist( 2): 3540  3552  3679  3752  3773  4257  4267  4393  4398  4474
    Sublist( 3): 4575  4594  4750  4915  4948  4972  4979  5021  5053  5128  5262  5428  5532
    Sublist( 4): 5627  5678  5694  6172  6198  6356  6517  6568
    Sublist( 5): 7019  7203  7290  7302  7314  7360  7585  7741
    Sublist( 6): 7764  7830  7837  7979  8210  8240  8398  8446  8594  8602  8741
    Sublist( 7): 8869  8945  9067  9107  9107  9434  9657  9838  9858  9927

 +++ Writing back to A
    1251  1309  1374  1474  1913  1977  2005  2048  2411  2442
    2751  2923  2968  3031  3231  3540  3552  3679  3752  3773
    4257  4267  4393  4398  4474  4575  4594  4750  4915  4948
    4972  4979  5021  5053  5128  5262  5428  5532  5627  5678
    5694  6172  6198  6356  6517  6568  7019  7203  7290  7302
    7314  7360  7585  7741  7764  7830  7837  7979  8210  8240
    8398  8446  8594  8602  8741  8869  8945  9067  9107  9107
    9434  9657  9838  9858  9927
```