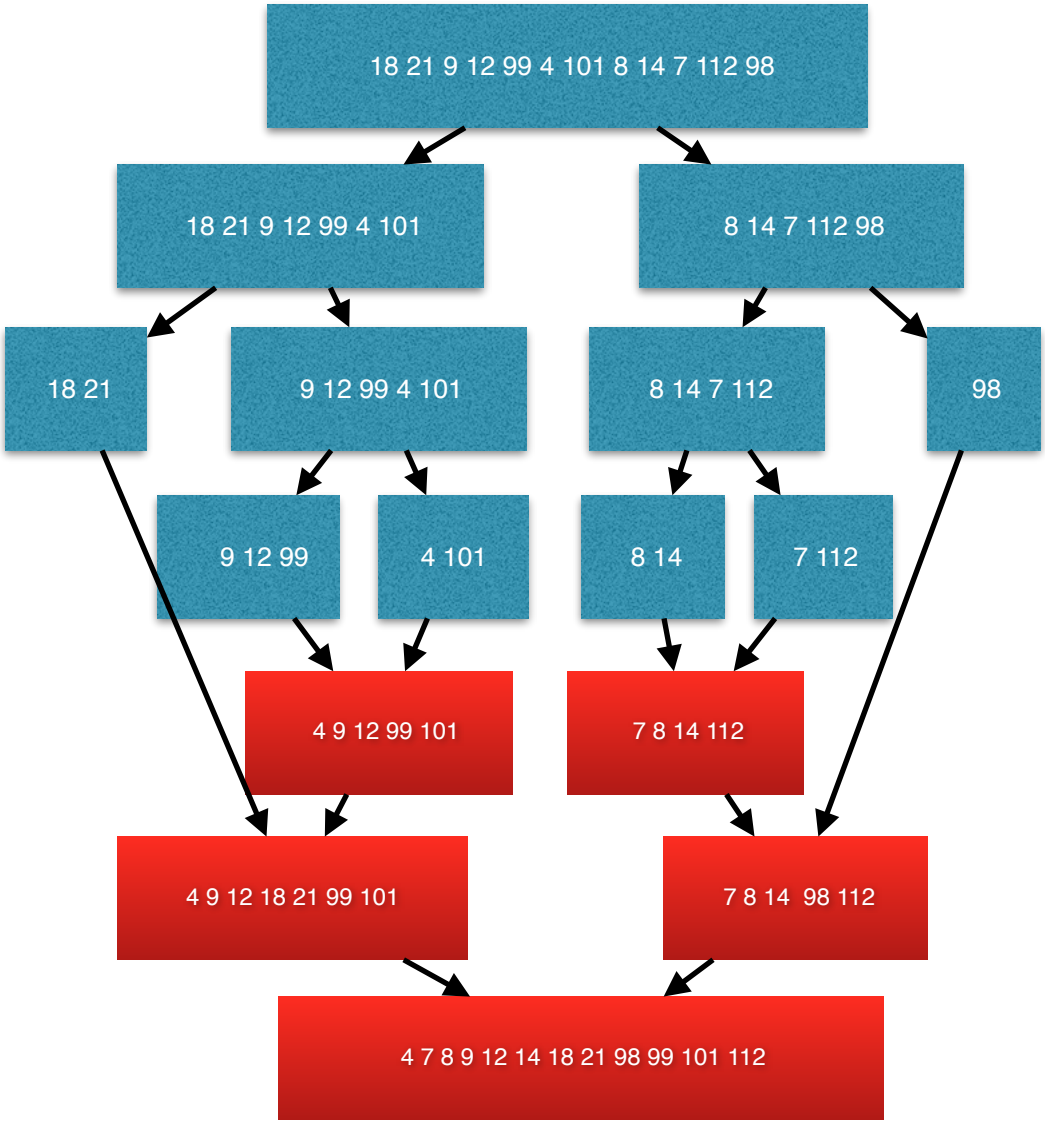
CSC 225 (Fall 2016): Algorithms and Data Structures I
Assignment 2—programming

Write a program that, given a sequence S of n numbers, will sort these numbers in *increasing* order using the following variant of mergesort, called chunk mergesort.

In chunk mergesort, the recursive algorithm takes advantage of already sorted "chunks" in S, that is of subsequences of consecutive elements in S that are in increasing order. More specifically, chunk mergesort divides the sequence S into sequences $S1$ and $S2$ as follows (*chunk divide*): After dividing, as with traditional mergesort, elements are divided into sequences $S1$ and $S2$. In c*hunk* divide when dividing the sequence into two, instead of containing about the same number of elements each, we divide according to the number of chunks. That is in $S1$ and $S2$ the number of chunks is to be about the same each.

Chunk mergesort is illustrated with the following example. Note that the input sequence, S = 18, 21, 9, 12, 99, 4, 101, 8, 14, 7, 112, 98, contains these six chunks: 18, 21; 9, 12, 99; 4, 101; 8, 14; 7, 112; and 98



Your task is to write a java program, stored in a file ChunkMergesort.java that contains the following functions: ChunkMergesort, which takes a list S as its only argument and returns a list. ChunkDivide, which takes list S, and integer c, the number of chunks that S contains, and returns lists S1 and S2. (In the case that c is odd, S1 shall contain more chunks than S2). Merge takes lists S1 and S2, and returns a (sorted) list. In addition, your program must output, in a file called comparisons.txt, all comparisons that take place in your program in order of occurrence.

Both, ChunkDivide and Merge are to run in linear time.

Use the main function in your code to test your implementation by getting test data or reading it from a file. It should also handle errors.

The attachment shows the output for comparisons.txt for above example.

comparisons.txt

18 21
21 9
9 12
12 99
99 4
4 101
101 8
8 14
14 7
7 112
112 98

18 21
21 9
9 12
12 99
99 4
4 101
101 8

18 21
21 9
9 12
12 99
99 4
4 101

18 21
21 9
9 12
12 99
99 4
18 21
21 9
9 12
12 99

18 21
21 9

18 9
18 12
18 99
21 99

9 4
9 101

12 101
18 101
21 101
99 101

8 14
14 7
7 112
112 98

4 7
9 7
9 8
9 14
12 14
18 14
18 98
21 98
99 98
99 112
101 112