// FILE: **TaskListWorkSheet.txt**
//
// To facilitate testing, I created a special InputStream file, **FakeInputStream.csv,** which has an
// additional field to use as a substitute for population for the priority value. These are small
// numbers, 1-72, corresponding to the smallest-to-largest populations. So the results will be
// the same as if population was used, it's just much easier to manually predict the correct
// output. NOTE: **The PROGRAM MUST USE InputStream.csv** NOT FakeInputStream.csv
//
// Heap is initialized when PQ object created (so BEFORE any BUILD is done).

build,continent,South America

// This adds 14 countries to heap with these FakePriorityValues (in this order of calling pq.add)
// 57, 2, 53, 12, 16, 63, 24, 43, 60, 54, 37, 18, 58, 71 for these countries:
// (I use only the 1st 3 letters for easier worksheet writing, program uses whole name)
// Ven, Fal, Ecu, Fre, Sur, Col, Uru, Bol, Arg, Chi, Par, Guy, Per, Bra
// This growing of the heap (with 14 pq.add's, so 14 heapInsert's), does WalkUp's 14 times.
// Show BOTH a logical view (i.e., a tree picture) AND a physical view (i.e., the array & N)
// so you practice how heapInsert really works.
//

// SCRATCH PAPER → A

//

arraySnapshot

// N = 14 and the array looks like this:

// 0	7
// 1	8
// 2	9
// 3	10
// 4	11
// 5	12
// 6	13

empty

// pq.empty is called ONCE - and that method calls heapDelete REPEATEDLY until N is 0. Be
// careful not to use actual N in a for-loop controller, since you'll be decrementing N til it hits 0.
// Use a while loop til heap is empty, or a for-loop which uses copyOfN instead of actual N to
// control it.
// Show both a logical view AND physical view of what's happening so you practice how
// heapDelete really works.
// Of course the predicted output (on the Log file) will end up being (since it's a MaxHeap):
// 71 63 60 58 57 54 53 43 37 24 18 16 12 2
//

// SCRATCH PAPER → B

//

arraySnapshot

// OK, so the array's now empty, with N=0

build,continent,North America

// This does pq.add 13 times with these FakePriorityValues:
// 9, 59, 10, 4, 72, 13, 69, 39, 23, 33, 29, 52, 38
// for these countries: Gre, Can, Ber, St., Uni, Bel, Mex, Hon, Pan, Nic, Cos, Gua, El
// Grow the heap. Just do the logical view (if you understand how algorithm works on the
// physical view, since that's what your program's actually using).
//

// SCRATCH PAPER → C

//

arraySnapshot

// N = 13 and the array looks like this:

// 0	7
// 1	8
// 2	9
// 3	10
// 4	11
// 5	12
// 6	

add,region,Western Europe

// This adds 9 Western Europe countries: 6, 17, 40, 41, 48, 66, 7, 55, 68
// which are: Lie, Lux, Swi, Aus, Bel, Fra, Mon, Net, Ger
// by calling pq.add 9 times. Add these to the current heap which had N=13 before these adds.
// Show the new tree as a tree.
//

// SCRATCH PAPER → D

//
// At this point N is 13+9 = 22 after the above adds.

add,region,Nordic Countries

// This adds 7 Nordic Countries countries: 44, 8, 3, 14, 32, 34, 35
// which are: Swe, Far, Sva, Ice, Nor, Fin, Den
// by calling pq.add 7 times. Add these to the current heap which had N=22 before these adds.
// Show the tree view.
// (Maybe you also want to see what the predicted array view is supposed to look like).
//

// SCRATCH PAPER → E

//
// At this point N is 22+7 = 29 (before doing the remove, 20).

remove,20

// This calls pq.remove 20 times (which calls HeapDelete each time (i.e.,
// remove the root, replace it with the last element, decrement N, call WalkDown).
// This process results in removing the 20 biggest ones, in descending order.
// Use the tree from picture **E** to determine which they are.
// At this point N is 29 – 20 = 9.

// Show the tree view.
//
// SCRATCH PAPER → F
//
arraySnapshot
// show the physical array view.
// N = 9 and the array looks like this:
// 0 5
// 1 6
// 2 7
// 3 8
// 4
add,region,British Islands
// This adds 2 countries: 27, 67 for Ire, Uni.
// So N = 11 now.
// Show the tree view.
//
// SCRATCH PAPER → G
//
remove,2
// Removes the 2 biggest. So N is now 11 – 2 = 9.
// Use the tree from picture ***G*** to determine 1) which they are and
// 2) what the tree looks like after.
//
// SCRATCH PAPER → H
//
remove,2
// Removes the 2 biggest. So N is now 9 – 2 = 7.
// Use the tree from picture **H** to determine 1) which they are and
// 2) what the tree looks like after.
//
// SCRATCH PAPER → I
//
add,region,Southern Europe
// This adds 14 countries: 25, 11, 50, 1, 51, 45, 15, 65, 20, 21, 31, 5, 28, 62
// for: alb, and, gre, vat, yug, por, mal, ita, slo, mac, cro, san, bos, spa
// So N is 7 + 14 = 21 now.
// Show the tree view.
//
// SCRATCH PAPER → J
//
add,region,Baltic Countries
// This adds 3 countries: 19, 22, 26 for: est, lat, lit
// So N is 21 + 3 = 24 now.
// Show the tree view.
//
// SCRATCH PAPER → K
//
remove,10
// Removes the 10 biggest. So N is now 24 – 10 = 14.
// Use the tree from picture **K** to determine which they are and what the tree looks like.
//
// SCRATCH PAPER → L
//
add,region,Eastern Europe
// This adds 10 countries: 30, 36, 70, 42, 61, 46, 47, 49, 56, 64
// for: mol, slo, rus, bul, pol, hun, bel, cze, rom, ukr
// So N is 14 + 10 = 24 now.
// Show the tree view.
//
// SCRATCH PAPER → M
//
arraySnapshot
// show the physical array view.
// N = 24 and the array looks like this:
// 0 12
// 1 13
// 2 14
// 3 15
// 4 16
// 5 17
// 6 18
// 7 19
// 8 20
// 9 21
// 10 22
// 11 23
Empty
// pq.empty is called ONCE - and that method calls heapDelete REPEATEDLY until N is 0.
// Use the tree from picture **M** to determine which they are in what order.
//
// SCRATCH PAPER → N
//