Feedback — Mergesort

You submitted this quiz on **Sun 21 Feb 2016 10:03 AM PST**. You got a score of **3.00** out of **3.00**.

To specify an array or sequence of values in an answer, separate the values in the sequence by whitespace. For example, if the question asks for the first ten powers of two (starting at 1), then the following answer is acceptable:

```
1 2 4 8 16 32 64 128 256 512
```

If you wish to discuss a particular question and answer in the forums, please post the entire question and answer, including the seed (which can be used by the course staff to uniquely identify the question) and the explanation (which contains the correct answer).

Question 1

(seed = 217207)
Give the array that results immediately after the 7th call (and return) from merge() when top-down mergesorting the following array of size 12:

```
80 14 83 49 68 15 13 34 30 43 33 29
```

Your answer should be a sequence of 12 integers, separated by whitespace.

You entered:

14 15 49 68 80 83 13 30 34 43 33 29

Your Answer		Score	Explanation
14 15 49 68 80 83 13 30 34 43 33 29	✔	1.00	
Total		1.00 / 1.00	

Question Explanation

The correct answer is: 14 15 49 68 80 83 13 30 34 43 33 29

Here is the array immediately after each call to merge():

```
                80 14 83 49 68 15 13 34 30 43 33 29
merge(0, 0, 1):  14 80 83 49 68 15 13 34 30 43 33 29
merge(0, 1, 2):  14 80 83 49 68 15 13 34 30 43 33 29
merge(3, 3, 4):  14 80 83 49 68 15 13 34 30 43 33 29
merge(3, 4, 5):  14 80 83 15 49 68 13 34 30 43 33 29
merge(0, 2, 5):  14 15 49 68 80 83 13 34 30 43 33 29
merge(6, 6, 7):  14 15 49 68 80 83 13 34 30 43 33 29
merge(6, 7, 8):  14 15 49 68 80 83 13 30 34 43 33 29
```

Question 2

(seed = 192418)
The column on the left contains an input array of 12 strings to be sorted; the column on the right contains the strings in sorted order; each of the other 4 columns contains the array at some intermediate step during either top-down or bottom-up mergesort (with different columns potentially corresponding to different algorithms).

mist	bone	bone	mist	buff	bone
palm	buff	buff	palm	herb	buff
herb	herb	herb	buff	mist	cafe
buff	lust	lust	herb	palm	dusk
lust	mist	mist	bone	bone	gold
bone	palm	palm	lust	lust	herb
rust	dusk	rust	ruby	ruby	lust
ruby	ruby	ruby	rust	rust	mist
dusk	rust	dusk	dusk	cafe	palm
gold	cafe	gold	gold	dusk	pink
pink	gold	pink	cafe	gold	ruby
cafe	pink	cafe	pink	pink	rust
----	----	----	----	----	----
0	?	?	?	?	3

Match up each column with the corresponding mergesorting algorithm from the given list:

0. Original input
1. Top-down Mergesort (standard recursive version)
2. Bottom-up Mergesort (nonrecursive version)
3. Sorted

You should use each choice at least once. Your answer should be a sequence of 6 integers between 0 and 3 (starting with 0 and ending with 3), separated by whitespace.

Hint: think about algorithm invariants. Do not trace code.

You entered:

```
0 1 1 2 2 3
```

Your Answer		Score	Explanation
0	✔	0.17	
1	✔	0.17	
1	✔	0.17	
2	✔	0.17	
2	✔	0.17	
3	✔	0.17	
Total		1.00 / 1.00	

Question Explanation

```
The correct answer is: 0 1 1 2 2 3

 0: Original input
 1: Top-down mergesort after 9 iterations
 1: Top-down mergesort after 5 iterations
 2: Bottom-up mergesort after forming subarrays of length 2
 2: Bottom-up mergesort after forming subarrays of length 4
 3: Sorted
```

Question 3

(seed = 972020)
Which of the following statements about mergesort are true? Check all that apply. Unless otherwise specified, assume that mergesort refers to the pure recursive (top-down) version of mergesort (with no optimizations), using the merging subroutine described in lecture.

Your Answer		Score	Explanation
☑ Any two items are compared with one another no more than once during bottom-up mergesort.	✔	0.20	Once two items are compared, they are merged into the same subarray. Only items in different subarrays can be compared.
☑ When merging two subarrays, the main reason for taking equal keys from the left subarray before the right subarray is to ensure stability.	✔	0.20	This is precisely the reason.
☐ Suppose we have a sorting algorithm that in addition to regular compares, is also allowed super-compares: take three keys and return those three keys in sorted order. Then, any compare-based sorting algorithm requires at least lg (N!) compares or super-compares (in the worst case) to sort an array of N items.	✔	0.20	Similar to the lower bound argument with 2-way compares, but now the height of the tree is at least log_6 (N!) since each node has as many as 6 children, corresponding to the 3! possible outcomes for each super-compare.

☐ `The number of compares in bottom-up mergesort depends only on the size of the array N (and not on the items in the array).`	✔	0.20	The number of compares ranges from ~ 1/2 N lg N (sorted array) to ~ N lg N (random array).
☑ `It is straightforward to adapt bottom-up mergesort to stably sort a singly-linked list of N items using a linearithmic number of compares and only constant extra memory (other than the input linked list).`	✔	0.20	The merging step can be done with a linear number of compares using only constant extra space.
Total		1.00 / 1.00	