## Homework M9: Sorting and Searching

**Due** Apr 11 at 11:59pm **Points** 100 **Questions** 45 **Available** until May 30 at 11:59pm **Time Limit** None

## Instructions

Review the **Homework FAQ** page for details about submitting homework. In this homework, you will:

- evaluate sorts
- trace sorts
- write code related to sorting
- · evaluate searches
- trace searches



Below is the driver/tester program. I strongly recommend using this to test your code before submitting. You can ignore the *test methods* at the end of the file.

<u>SearchSortHomeworkFiles.zip</u> <u>↓</u> (https://ccsf.instructure.com/courses/47904/files /7254665/download?download\_frd=1)

This quiz was locked May 30 at 11:59pm.

## **Attempt History**

	Attempt	Time	Score
LATEST	Attempt 1	21,408 minutes	107 out of 100

Score for this quiz: **107** out of 100

Submitted Apr 8 at 10:25pm

This attempt took 21,408 minutes.

## **Coding Questions**

The *sortedness* of a dataset is the degree to which that dataset is sorted, ranging from 0 %to 100%. You will write methods to measure the sortedness of data stored in an array and in a chain of linked nodes.

Sortedness can be measured in different ways. We will use an approach based on *inversions*. An inversion is two elements such that the index of element1 is less than the index of element2, but the value of element1 is greater than the value of element2.

For the homework, compare neighbor elements only. The degree of sortedness is the percentage of neighbors that are in sorted order (meaning they are **not** inversions). The percentage should be a decimal value between 0 and 1.

For our purposes, we are considering only ascending order (smallest to largest) as being sorted.

- Example: [2, 2, 4, 6, 7] has a 100% sortedness factor- all neighbor pairs of numbers are sorted.
- Example: [2, 2, 6, 4, 7] has a 75% sortedness factor- the 6-4 neighbor pair is not sorted, but all other neighbor pairs (2-2, 2-6, and 4-7) are sorted.
- Example: [4, 5, 2] has a 50% sortedness factor- the 4-5 neighbor pair is sorted and the 5-2 neighbor pair is not.
- Example: [7, 6, 4, 2, 1] has a 0% sortedness factor- no neighbor pairs are sorted in ascending order.

See the homework driver for more examples and test cases.

Question 1 10 / 10 pts

Write a method to measure sortedness of an array of Comparable elements. The method header is:

```
public static double sortedness(Comparable[] array)
```

## Your Answer:

Question 2 20 / 10 pts

Write a method to measure sortedness of a collection of linked nodes that hold Comparable data.

public static double sortedness(Node<Comparable> node)

**Optional Extra Credit:** +10 extra credit points if your solution is recursive, instead of iterative.

## Your Answer:

```
// sortedness node
public static double sortedness(Node<Comparable> node) {
    double sortedness = 1;
    double totalPairs = 0;
    double orderedPairs = 0;

    Node<Comparable> currentNode = node;

// get total pairs (n - 1)
    while (currentNode != null) {
    totalPairs++;
    currentNode = currentNode.next;
    }
```

```
// reset currentNode to firstNode
    currentNode = node;
    if (currentNode != null) {
        orderedPairs += sortednessHelper(currentNode);
        return orderedPairs / (totalPairs - 1);
    return sortedness;
}
// helper method
private static double sortednessHelper(Node<Comparable> node) {
    double orderedPairs = 0;
    Node<Comparable> currentNode = node;
    if (currentNode != null && currentNode.next != null) {
        if (currentNode.data.compareTo(currentNode.next.data) <= 0) {</pre>
            orderedPairs++;
        orderedPairs = orderedPairs + sortednessHelper(currentNode.next);
    return orderedPairs;
}
```

## **Sort and Search Traces**

Question 3 8 / 8 pts

Trace Shell sort. Sort the data into ascending order (from smallest to largest).

- In addition to showing the contents of the array, list what the space/gap is for that part of the trace.
- Use the efficiency improvement described in the lecture nodes/videos (making sure the space is always odd).
- Review the <u>Example Sort Traces</u> page for an example of how to write this trace.

Dataset: [30, 20, 24, 16, 10, 25, 12, 14]

## Your Answer:

```
Dataset: [30, 20, 24, 16, 10, 25, 12, 14]
Space = 8/2 = 4 \rightarrow 4 \rightarrow 5
[30, 20, 24, 16, 10, 25, 12, 14]
    [30, 25]
    [25, 30]
[25, 20, 24, 16, 10, 30, 12, 14]
    [20, 12]
    [12, 20]
[25, 12, 24, 16, 10, 30, 20, 14]
    [24, 14]
    [14, 24]
[25, 12, 14, 16, 10, 30, 20, 24]
Space = 5/2 \rightarrow 2.5 \rightarrow 2 \rightarrow 3
[25, 12, 14, 16, 10, 30, 20, 24]
    [25, 16, 20]
    [16, 20, 25]
[16, 12, 14, 20, 10, 30, 25, 24]
    [12, 10, 24]
    [10, 12, 24]
[16, 10, 14, 25, 12, 30, 20, 24]
Space = 3/2 -> 1.5 -> 1 "aka regular insertion sort"
[16, 10, 14, 25, 12, 30, 20, 24]
[10, 12, 14, 16, 20, 24, 25, 30]
```

### 8 / 8 pts **Question 4**

Trace merge sort. Sort the data into ascending order (from smallest to

## largest).

- Split down to 1-element arrays.
- Show how the arrays are split and how the arrays are merged.
- Review the <u>Example Sort Traces</u> page for an example of how to write this trace.

Dataset: [37, 23, 28, 21, 12, 34, 15, 19]

## Your Answer:

The following questions ask about tracing a binary search. Use one of the two binary search algorithms given in the Searches.java file included in the <u>lecture code files</u>  $\downarrow$  (https://ccsf.instructure.com/courses/47904/files/7254596/download?download\_frd=1). (You can use either- you can choose.) The code is also posted below for reference.

To trace the binary search, figure out the first, last, and mid for **each pass** through the loop or recursive method call. You can review the **Example Binary Search Trace** page for examples of what I mean. The questions below will ask you to input the value of **mid** for each pass or recursive method call.

If you aren't sure how to answer these questions, post to the discussion board!

7 of 31

Use one of these methods for your trace.

```
public static int binarySearchIterative(int[] numbers,
int target) {
   boolean found = false;
   int first = 0;
   int last = numbers.length - 1;
   while (first <= last && !found) {</pre>
      int mid = (first + last) / 2;
      if (numbers[mid] == target) {
         targetLocation = mid;
         found = true;
      } else if (numbers[mid] < target) {</pre>
         first = mid + 1;
      } else { // numbers[mid] > target
         last = mid - 1;
   return targetLocation;
public static int binarySearchRecursive(int[] numbers,
int target) {
   return binarySearchRecursiveHelper(numbers, target,
0, numbers.length - 1);
private static int binarySearchRecursiveHelper(int[] nu
mbers, int target, int first, int last) {
   int mid = ((last - first) / 2) + first;
   if (first > last) {
      return -1; // indices cross over
   } else if (target == numbers[mid]) {
      return mid; // we found it!
   } else if (target < numbers[mid]) {</pre>
      return binarySearchRecursiveHelper(numbers, targe
t, first, mid - 1);
  } else { // target > numbers[mid]
      return binarySearchRecursiveHelper(numbers, targe
t, mid + 1, last);
}
```

Trace binary search on the sorted dataset below.

For these questions, mid is the **index** of the array, not the value at that index.

$$target = 15$$

index:	0	1	2	3	4	5	6	7	8	9	10	11
value:	12	15	19	23	25	27	36	39	47	58	62	67

	Question 5	2 / 2 pts
	What is the value of <b>mid</b> after the first loop or recursive call?	
Correct!	5	
orrect Answe	rs 27 (with margin: 0) 5 (with margin: 0)	

	Question 6 2 / 2 pts	
	What is the value of <b>mid</b> after the <b>second</b> loop or recursive call?  (If there is no second loop/recursive call, enter -1 for your answer.)	
Correct!	2	

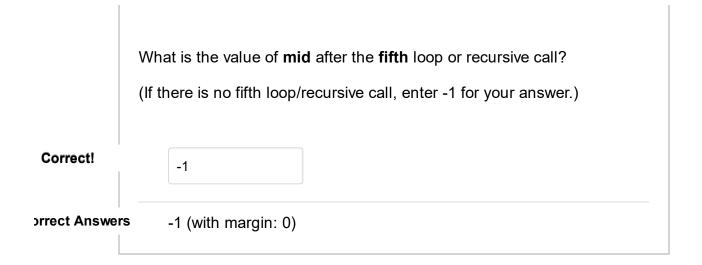
orrect Answers	19 (with margin: 0)	
	2 (with margin: 0)	

	Question 7	2 / 2 pts
	What is the value of <b>mid</b> after the <b>third</b> loop or recursive call?  (If there is no third loop/recursive call, enter -1 for your answer.	)
Correct!	0	
orrect Answer	s 0 (with margin: 0) 12 (with margin: 0)	

	Question 8	2 / 2 pts
	What is the value of <b>mid</b> after the <b>fourth</b> loop or recur (If there is no fourth loop/recursive call, enter -1 for you	
Correct!	1	
orrect Answers	1 (with margin: 0) 15 (with margin: 0)	

Question 9	2 / 2 pts

10 of 31



Trace binary search on the sorted dataset below.

For these questions, mid is the **index** of the array, not the value at that index.

$$target = 33$$

index:	0	1	2	3	4	5	6	7	8	9	10	11
value:	12	15	19	23	25	27	36	39	47	58	62	67

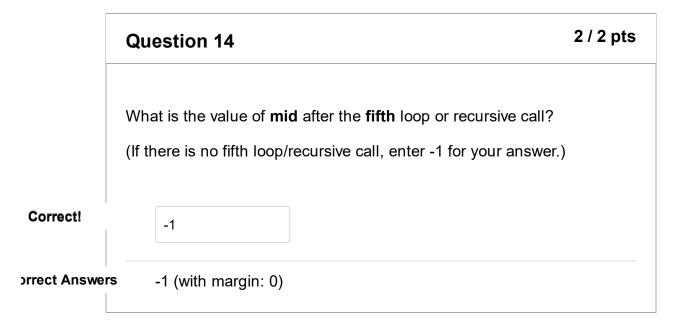
	Question 10	2 / 2 pts
	What is the value of <b>mid</b> after the <b>first</b> loop or recursive call?	
Correct!	5	
orrect Answers	5 (with margin: 0) 27 (with margin: 0)	

C	Question 11	2 / 2 pts
V	What is the value of <b>mid</b> after the <b>second</b> loop or re	ecursive call?
(1	If there is no second loop/recursive call, enter -1 for	r your answer.)
Correct!		
Correcti		
Correcti	8	
orrect Answers	8 (with margin: 0)	

	Question 12	2 / 2 pts
	What is the value of <b>mid</b> after the <b>third</b> loop or recursive (If there is no third loop/recursive call, enter -1 for your a	
Correct!	6	
orrect Answers	36 (with margin: 0) 6 (with margin: 0)	

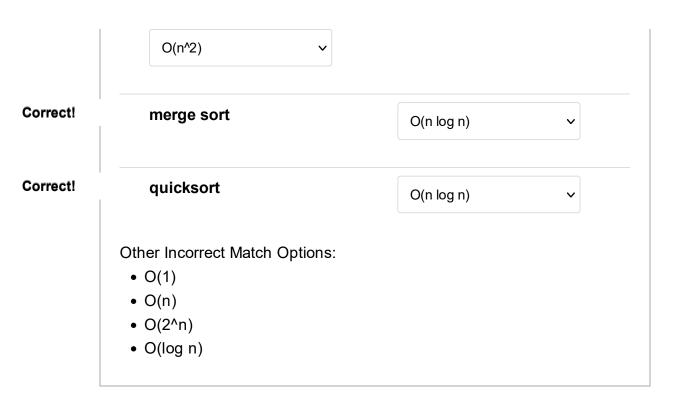
## Question 13 What is the value of mid after the fourth loop or recursive call? (If there is no fourth loop/recursive call, enter -1 for your answer.)





## **Multiple Choice Questions- Sorting**

	Question 15		2 / 2 pts
	What is the efficiency of each	sort? Use the average case	€.
Correct!	selection sort	O(n^2)	•
Correct!	insertion sort		



The next questions ask about traces of selection and insertion sort. For these questions, use **ascending order** for the sorts (smallest to largest).

	Question 16	2 / 2 pts
	Here is an unsorted array.	
	68 59 36 45 21	
	What are the contents of the array after the first pass of se	election sort?
	onone of these is correct	
Correct!	[21, 59, 36, 45, 68]	
	O [59, 68, 36, 45, 21]	

Correct!

O [21, 68, 59,	36, 45]	
O [21, 36, 45,	59, 68]	

# Here is an array after the first pass of selection sort. 3 9 6 5 What are the contents of the array after the next (the second) pass of selection sort? onone of these is correct [3, 6, 9, 5] [3, 5, 9, 6] [3, 9, 6, 5] [3, 5, 6, 9]

## Question 18 Here is an array after the second pass of selection sort. 17 24 36 45 Will there be a third pass of selection sort? If so, what are the contents of the array after that third pass?

	Yes there will be a third pass. But the correct contents of the array are not shown here.
Correct!	Yes. After the third pass, the array holds [17, 24, 36, 45]
	No, there will not be a third pass because the array is already sorted.
	O Yes. After the third pass, the array holds [17, 24, 45, 36]

	Question 19	/ 2 pts
	Here is an unsorted array.  46 31 25 22 17  What are the contents of the array after the <b>first pass</b> of <b>insertio</b>	n sort?
	O [17, 46, 31, 25, 22]	
	O none of these is correct	
	O [17, 22, 25, 31, 46]	
Correct!	[31, 46, 25, 22, 17]	
	O [17, 31, 25, 22, 46]	

Question 20	2 / 2 pts
Here is an array <b>after</b> the <b>first pass</b> of <b>insertion sort</b> .	

	3 8 7 4  What are the contents of the array after the <b>next (the second) pass</b> of insertion sort?
	O none of these is correct
	O [3, 4, 7, 8]
	O [3, 4, 8, 7]
	O [3, 8, 7, 4]
Correct!	[3, 7, 8, 4]

	Question 21	2 / 2 pts
	Here is an array <b>after</b> the <b>second pass</b> of <b>insertion sort</b> .  62 71 83 94  Will there be a <b>third pass</b> of insertion sort? If so, what are the of the array after that third pass?	contents
	Yes there will be a third pass. But the correct contents of the array not shown here.	are
Correct!	Yes. After the third pass, the array holds [62, 71, 83, 94]	
	O Yes. After the third pass, the array holds [62, 71, 94, 83]	
	<ul> <li>No, there will not be a third pass because the array is already s</li> </ul>	orted.

For the following questions, use the Quicksort code given in the textbook Chapter 16, Section 18 (Older Editions: Chapter 9, Section 18), which references the partition method shown in Section 17 of that chapter.

Do not use a different version of quicksort or partition.

```
partition(array, int first, int last)
   mid = index of middle entry
   sortFirstMiddleLast(array, first, mid, last)
   swap a[mid] and a[last-1]
   pivotIndex = last-1 // Line A: Start Here!
   pivotValue = array[pivotIndex]
   indexFromLeft = first+1
   indexFromRight = last-2
   done = false
   while(!done) { // Outer Loop Starts Here
      while(array[indexFromLeft] < pivotValue) {</pre>
         indexFromLeft++
      while(array[indexFromRight] > pivotValue) {
         indexFromRight--
      }
      if(indexFromLeft < indexFromRight) {</pre>
         swap array[indexFromLeft] and array[indexFromRig
ht] // Line B: In-Loop Swap
         indexFromLeft++
         indexFromRight--
      } else {
         done = true;
   } // Outer Loop Stops Here
   swap array[pivotIndex] and array[indexFromLeft] // Lin
```

```
e C: Out-Of-Loop Swap

pivotIndex = indexFromLeft
return pivotIndex // Line D: Return Statement
```

An array is passed to the partition method. This is is what the array looks like **after** the call to sortFirstMiddleLast and **after** the swap that comes next. In other words, this is what the array looks like when you reach the line labeled **Line A: Start Here!** 

index:	0	1	2	3	4	5	6	7
value:	15	24	25	11	10	26	19	32

After this line, and before starting the Outer Loop, the values are:

pivotIndex = 6

pivotValue = 19

indexFromLeft = 1

indexFromRight = 5

For the **first pass** through the outer while loop, what are the two indices swapped in **Line B: In-Loop-Swap**?

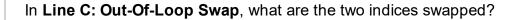
C	Question 22	1 / 1 pts
in	dexFromLeft	
Correct!	1	
rect Answers	24 (with margin: 0)	
	1 (with margin: 0)	

	Question 23	0 / 1 pts
	indexFromRight	
ou Answered	5	
orrect Answer	10 (with margin: 0) 4 (with margin: 0)	

For the **second pass** through the outer while loop, what are the two indices swapped in **Line B: In-Loop-Swap**?

Question 24	1 / 1 pts
indexFromLeft	

Correct!	2
orrect Answers	2 (with margin: 0) 25 (with margin: 0)
	Question 25 1/1 pts
	indexFromRight
Correct!	3
orrect Answers	3 (with margin: 0) 11 (with margin: 0)
	Question 26 0 / 1 pts
	Question 26 0 / 1 pts  Following this same trace, the outer while loop will be entered for third time.
orrect Answer	Following this same trace, the outer while loop will be entered for <b>third</b>
orrect Answer	Following this same trace, the outer while loop will be entered for <b>third</b> time.
	Following this same trace, the outer while loop will be entered for <b>third</b> time.  O True



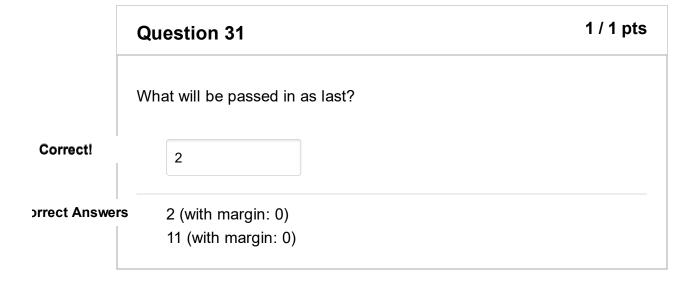
	Question 27	1 / 1 pts
	pivotlndex	
Correct!	6	
orrect Answer	6 (with margin: 0) 19 (with margin: 0)	

	Question 28	1 / 1 pts
	indexFromLeft	
Correct!	3	
orrect Answe	rs 3 (with margin: 0) 25 (with margin: 0)	

	Question 29	1 / 1 pts
	In Line D: Return Statement, what value is returned?	
Correct!	3	
orrect Answer	19 (with margin: 0)	

3 (with margin: 0)

## Using the quickSort method in Section 18, after the above pivotIndex is returned, what will be passed in as first in the next recursive call to quickSort? Correct! 0 15 (with margin: 0) 0 (with margin: 0)



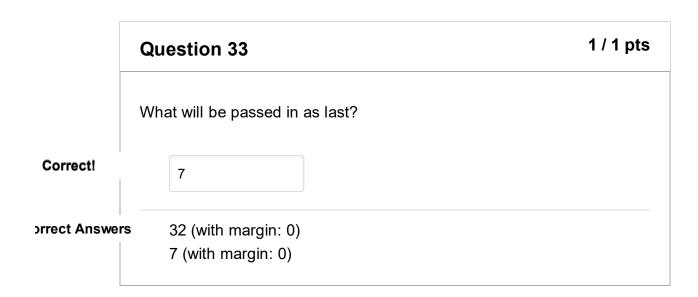
```
Question 32

What will be passed in as first in the second recursive call to quickSort?

Correct!

4
```

## 24 (with margin: 0) 4 (with margin: 0)



## Question 34 1 / 1 pts

One pass of the partition method is performed with the partition in index 4 (the value 50).

Could the array below be the result of one pass of partition?

index:	0	1	2	3	4	5	6	7	8
value:	2	14	28	62	50	48	63	77	81

## Correct!

No

Yes

No because the 62 is to the left of 50 and the 48 is to the right of the 50.

1 / 1 pts

## Question 35 1 / 1 pts

One pass of the partition method is performed with the partition in index 4 (the value 50).

Could the array below be the result of one pass of partition?

index:	0	1	2	3	4	5	6	7	8
value:	3	15	7	19	50	68	92	51	72

O No

## Correct!

Yes

**Question 36** 

All values to the left of 50 are less than 50 and all values to the right of 50 are greater than 50.

## **Multiple Choice Questions- Searching**

	You can use linear search on any kind of data (array-based, linked nodes, sorted, or unsorted).
Correct!	True
	O False

## You can use binary search on any kind of data (array-based, linked nodes, sorted, or unsorted). True Binary search cannot be performed on unsorted data. Binary search should not be performed on linked nodes because there is no direct access to any given node.

Question 38	1 / 1 pts

Correct!

Binary search is more efficient that linear search. In other words, binary search has a **lower order of growth (Big-O)** compared to linear search.

True

False

Binary search is O(log n) and linear search is O(n). Logarithmic growth is more efficient than linear growth.

# In all cases, with any given sorted dataset, binary search will always make fewer comparisons than linear search. True Binary search is more efficient than linear search. But there will certainly be data examples where linear search will find the target with fewer comparisons. Remember that efficiency doesn't describe the time an algorithm takes to run. It describes how the solution time increases as the problem size grows.

The following questions ask about tracing a linear search. Use the

012345

Correct!

"optimized" version that stops when the target is found.

The answers show the **indices** of the array that are visited for the search.

## 2 / 2 pts **Question 40** What indices will be visited for the unsorted data below? target = 165 2 7 index: 0 1 6 8 value: 12 14 | 18 | 21 29 17 16 15 | 13 012345678 onone of these is correct 0 1 2 3 4 5 6

Question	41										2/2	2 pts
What indices	s will be	visite	ed fo	or the	e un	sorte	ed da	ata b	elov	v?		
				tar	get	= 92	2					
	index:	0	1	2	3	4	5	6	7	8		
	value:	12	14	18	21	29	17	16	15	13		
				ı	,	•						
O 0												

## Correct!

•	0 1 2 3 4 5 6 7 8
0	none of these is correct
0	0 1 2 3 4 5 6 7 8 9

## Question 42 2 / 2 pts

What indices will be visited for the unsorted data below?

index:	0	1	2	3	4	5	6	7	8
value:	12	14	18	21	29	17	16	15	13

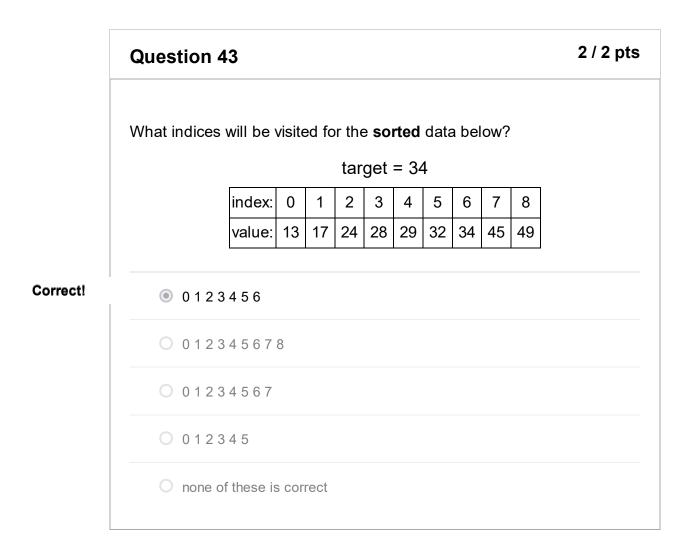
- O none of these is correct
- 0123456789
- 0

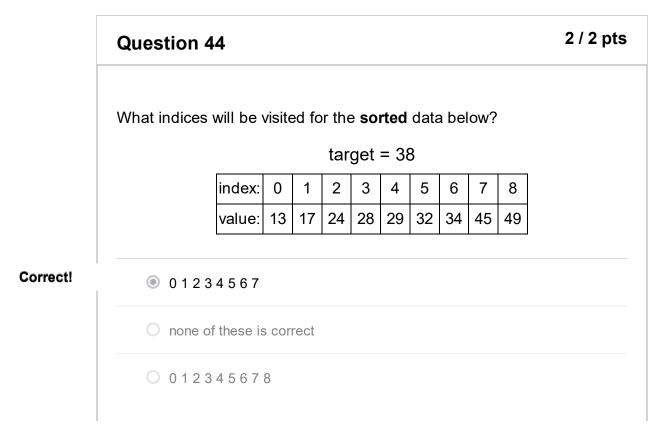
## Correct!

0 1 2 3 4 5 6 7 8

The following questions ask about tracing a linear search on **sorted** data. Use the "optimized" version that stops when the target is found and the version that stops if it's no longer possible for a target to be in the array.

The answers show the **indices** of the array that are visited for the search.





O 1 2 3 4 5 6	

	Question 4	15										2 / 2 pt
	What indices will be visited for the <b>sorted</b> data below?											
					tarç	get :	= <b>-</b> 2	1				
		index:	0	1	2	3	4	5	6	7	8	
		value:	13	17	24	28	29	32	34	45	49	
	0 0 1	4567	8									
	0 012											
	O none o	f these is	cor	rect								
orrect!	<b>o</b> 0											
	O no indi	ces will b	e vis	sited								

Quiz Score: 107 out of 100

6/3/2022, 9:11 AM 31 of 31