

Date	<i>April 27, 2023</i>
Course	<i>CS102 - DSA, Spring 2023</i>
Exam	<i>Bonus Exam, L4</i>
Time Limit	<i>60 minutes</i>
Instructor	<i>Mohammad Salman</i>

Instructions for students

- This document contains 12 pages (numbered at the top-right of each page).
- There are four questions.
- None of them are optional.

Question 1 - Recursion / Merge Sort	<i>7 marks</i>
Question 2 - Sorting	<i>9 marks</i>
Question 3 - Hashing	<i>1 mark</i>
Question 4 - Hashing	<i>14 marks</i>
Total	<i>31 marks</i>

Name: _____

Section: **L4**

DO NOT TURN THIS PAGE UNTIL INSTRUCTED

Question 1 - Recursion / Merge Sort [Marks = / 7]

Merge Sort is a recursive sorting algorithm. You are given the following array, data.

[7]

data = [1, 2, 3, 4, 5, 6, 7, 8]

Below is a **merge sort recursion tree** that shows the various function calls that take place.

Fill in the boxes with the appropriate subarrays on which the recursion calls are made. Some calls are already shown for your understanding.

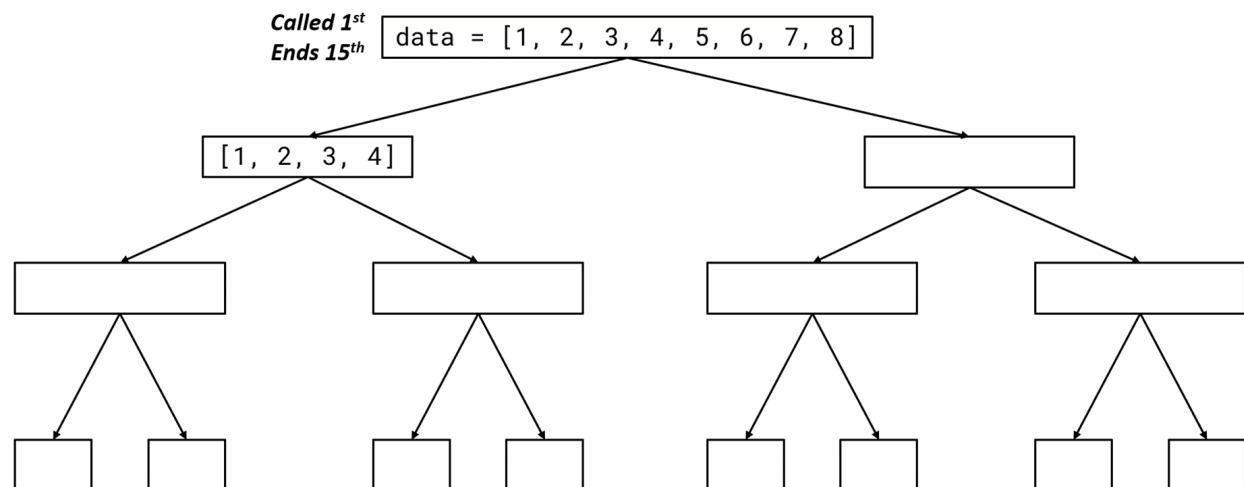
Additionally, indicate when a recursive call is made, and when it finishes execution. You can do this by writing,

*Called 1st, 2nd, etc., and
Ends 1st, 2nd, etc.,*

next to each call.

For example, the function call denoted by the root of the recursion tree gets called 1st and ends last (15th).

Hint: Recall the order of recursive calls in Merge Sort.



Question 2 - Sorting [Marks = / 9]

The sorting algorithms that we have studied thus far are based on some sorting logic which dictates the entire sorting process.

You are given an array of integers that is unsorted. In each of the three parts below, you are given a visualization of the sorting process for this same array.

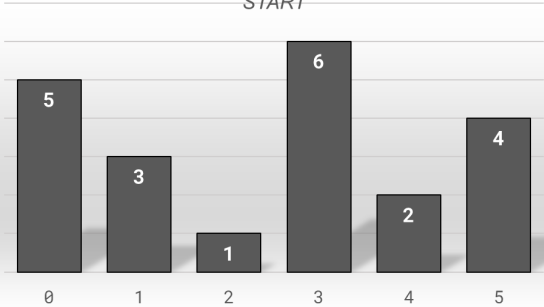
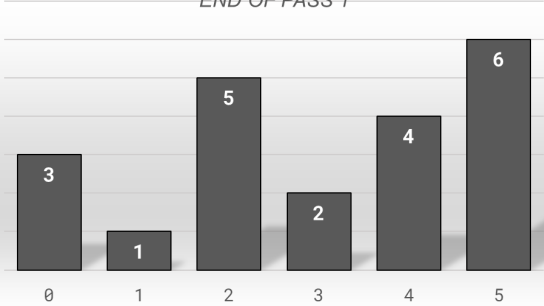
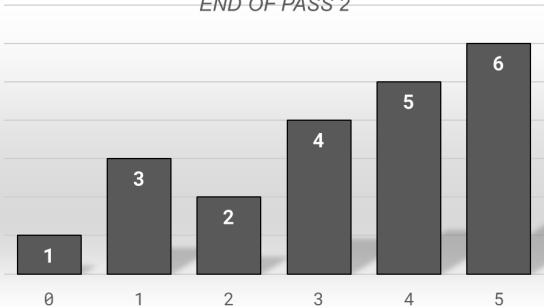
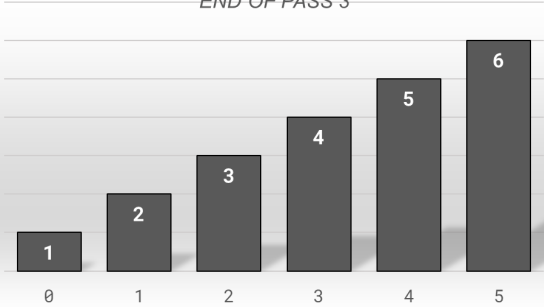
Based on the visualizations, state which sorting algorithm has been implemented to sort the array. Briefly explain your reasoning.

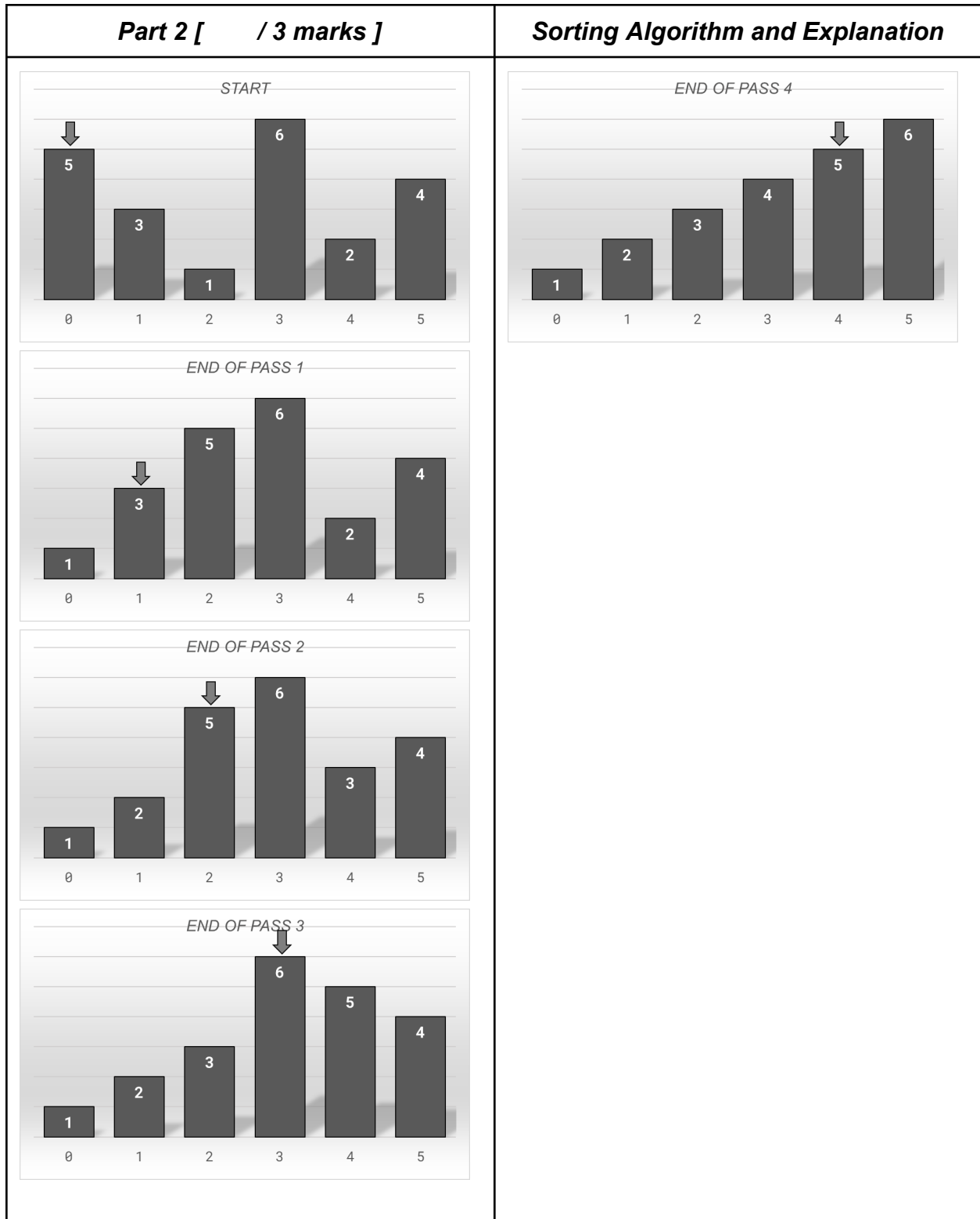
`arr = [5, 3, 1, 6, 2, 4]`

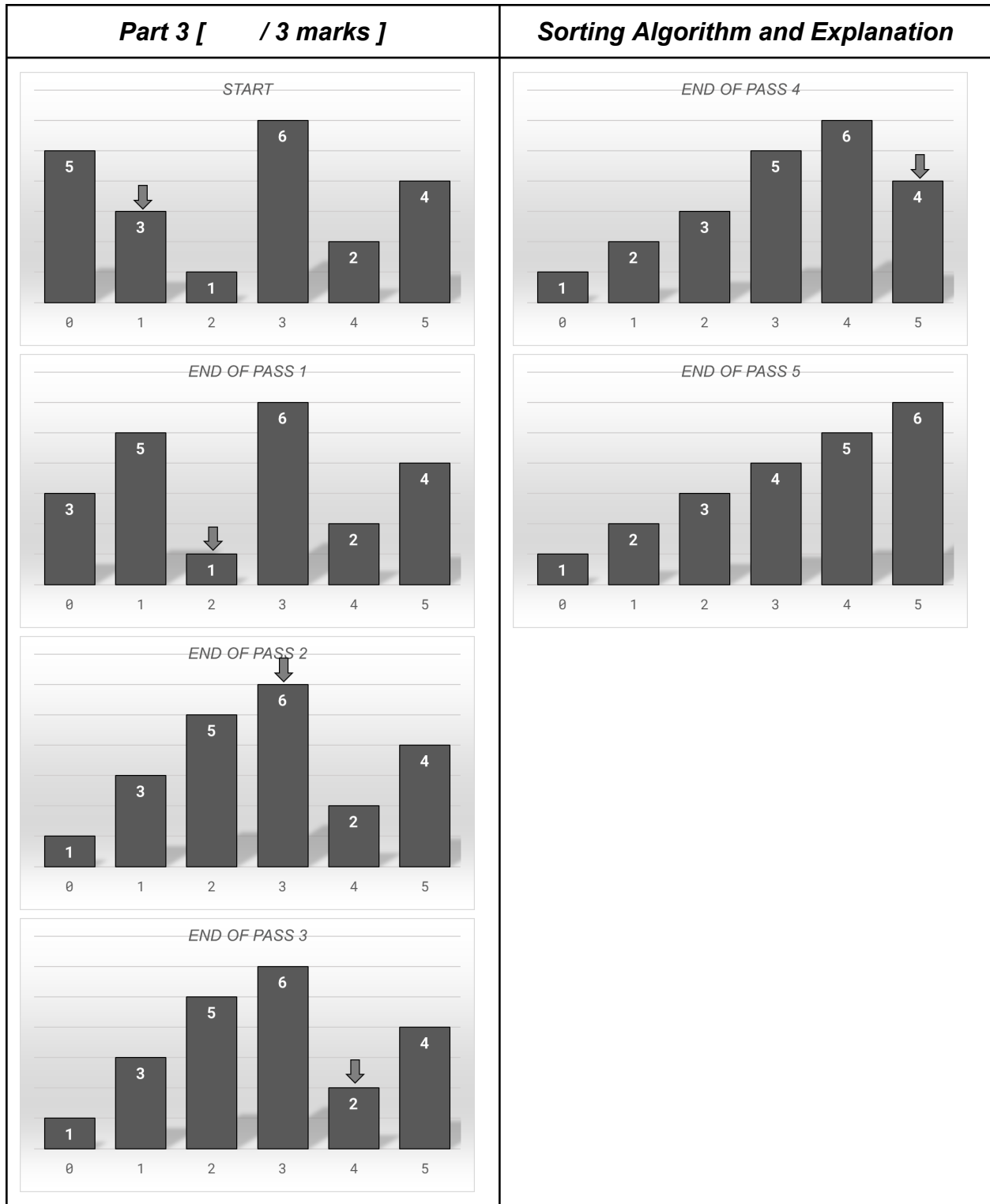
Note

- All of the steps are **not** shown; only the major passes or the major steps are shown in each visualization.
- Assume that `arr` has to be sorted in an **ascending order**.

BLANK SPACE
QUESTION CONTINUED ON NEXT PAGE

Part 1 [/ 3 marks]	Sorting Algorithm and Explanation																																																								
<div data-bbox="207 275 781 621"><p>START</p><table border="1"><thead><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>5</td></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>2</td></tr><tr><td>5</td><td>4</td></tr></tbody></table></div> <div data-bbox="207 632 781 978"><p>END OF PASS 1</p><table border="1"><thead><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>3</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>5</td></tr><tr><td>3</td><td>2</td></tr><tr><td>4</td><td>4</td></tr><tr><td>5</td><td>6</td></tr></tbody></table></div> <div data-bbox="207 1020 781 1367"><p>END OF PASS 2</p><table border="1"><thead><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>4</td></tr><tr><td>4</td><td>5</td></tr><tr><td>5</td><td>6</td></tr></tbody></table></div> <div data-bbox="207 1377 781 1724"><p>END OF PASS 3</p><table border="1"><thead><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>2</td></tr><tr><td>2</td><td>3</td></tr><tr><td>3</td><td>4</td></tr><tr><td>4</td><td>5</td></tr><tr><td>5</td><td>6</td></tr></tbody></table></div>	Index	Value	0	5	1	3	2	1	3	6	4	2	5	4	Index	Value	0	3	1	1	2	5	3	2	4	4	5	6	Index	Value	0	1	1	3	2	2	3	4	4	5	5	6	Index	Value	0	1	1	2	2	3	3	4	4	5	5	6	
Index	Value																																																								
0	5																																																								
1	3																																																								
2	1																																																								
3	6																																																								
4	2																																																								
5	4																																																								
Index	Value																																																								
0	3																																																								
1	1																																																								
2	5																																																								
3	2																																																								
4	4																																																								
5	6																																																								
Index	Value																																																								
0	1																																																								
1	3																																																								
2	2																																																								
3	4																																																								
4	5																																																								
5	6																																																								
Index	Value																																																								
0	1																																																								
1	2																																																								
2	3																																																								
3	4																																																								
4	5																																																								
5	6																																																								





Question 3 - Hashing [Marks = / 1]

<p>A variation of linear probing is that instead of doing the following (assuming a collision occurs at $A[i]$),</p> $A[i] \rightarrow A[i+1] \rightarrow A[i+2] \rightarrow A[i+3] \rightarrow A[i+4] \rightarrow \dots$ <p>you probe the hash table in alternating directions as shown below (again assuming a collision occurs at $A[i]$),</p> $A[i] \rightarrow A[i+1] \rightarrow A[i-1] \rightarrow A[i+2] \rightarrow A[i-2] \rightarrow \dots$ <p>Assume that in both cases, a cyclic behavior exists so that there is no out of bounds error.</p> <p>Which one of the following options is true for this variant of linear probing?</p> <ol style="list-style-type: none"> It will cause primary clustering. It will cause secondary clustering. It will cause neither primary clustering nor secondary clustering. This variant violates the definition of a collision resolution technique. 	[1]
--	-----

Question 4 - Hashing [Marks = / 14]

<p>You are given a hash table of size = 6. You are required to update the hash table based on the operations that will be mentioned later. Note that the items to be added to the hash table are just integers, and not key-value pairs. Therefore, you can consider the integers as keys.</p> <p>Please keep the following information in mind while performing the operations.</p> <ul style="list-style-type: none"> Place the items using the following hash function. $h(\text{item}) = (\text{absolute value of item}) \% (\text{table size})$ You are given a list of integers. You need to insert the items starting first from index 0, then index 1, and so on. $\text{items} = [2, 5, -2, 9, 10, -8]$ 	
---	--

- In case of collisions, use the standard **linear probing** technique.

- If adding an item to the hash table will cause the following,

$$\lambda > \frac{2}{3}, \text{ where } \lambda \text{ is the load factor,}$$

then increase the size of the table to **10**, and transfer all **existing items** to the newly resized hash table **appropriately**.

- Note that in your working, you would have to extend the already drawn table when you encounter this situation.

For each operation mentioned below, draw the state of the hash table **after** you have performed that particular update.

Show your working (including any collision resolutions, other steps that you may perform, etc.).

1. **Add 2** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

2. **Add 5** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

3. **Add -2** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

4. **Add 9** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

5. **Delete -2** from the hash table.

Explain/show the steps you must take to get to -2 and then perform the deletion appropriately.

0	1	2	3	4	5

computed index =

load factor =

6. **Search for 9** in the hash table.

Explain and show the steps you must take to get to 9.

0	1	2	3	4	5

computed index =

load factor =

7. **Add 10** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

8. **Add -8** to the hash table.

0	1	2	3	4	5

computed index =

load factor =

END OF EXAM QUESTIONS

EXTRA PAGE

EXTRA PAGE

Merge Sort

Consider this question as part two for **Question 1** in this document.

Based on the recursion tree you drew and/or recalling the pseudocode for Merge Sort, write down the recurrence relation for Merge Sort.

Hint:

$$T(n) = \begin{cases} \text{base case}, & n = 1 \\ \text{recursive part} + \text{non recursive part}, & n > 1 \end{cases}$$

Based on the recursion tree that you completed, or by using the recurrence relation above, determine the time complexity of Merge Sort.

Merge Sort [Marks = / 7]

<p>Assume that you are at the final step of merge sort, and you are about to merge the two halves of the input array (which at this point are already sorted).</p> <p>For the following two situations, perform the merging process. Show the intermediate states. Assume that the final array is named main.</p> <p><u>Hint:</u> <i>Keep track of the number of comparisons you make while merging. This will help you in the last part of this question.</i></p>	
<p>a. left_half = [1, 2, 3, 4], right_half = [5, 6, 7, 8]</p>	[2]
<pre>main = [1] main = [1, 2] main = [1, 2, 3]</pre> <p>And so on...</p>	
<p>b. left_half = [1, 3, 5, 7], right_half = [2, 4, 6, 8]</p>	[2]
<p>Based on the merging that you did above, comment on which merging required a lesser number of comparisons.</p> <p>Provide an explanation on why this happened, if it happened.</p>	[3]

--	--

Hashing [Marks = / 3]

<p>You are given the following hash function which you should use to add items to your hash table.</p> $h(key) = (\frac{key}{RNG(key)}) \% tableSize$ <p><i>RNG()</i> is a pseudo-random number generator based on whatever input you provide to it. You should assume this generates a sequence of numbers dependent on the input.</p> <p>In the hash function above, you pick the very first element that <i>RNG()</i> produces, and use that to compute the home address.</p> <p>For collision resolution, do the following. If a collision occurs, you can pick the next number in the sequence generated by <i>RNG()</i>, and recompute $h(key)$, and so on.</p> <p>For this combination of hash function and collision resolution,</p> <ol style="list-style-type: none"> Is the computation of an index for a key deterministic? What type of clustering will be caused because of the collision resolution technique, if any? Explain. 	[3]

**Part 2****Sorting Algorithm and Explanation [3]**