Lab # 4

Problem statement

Assume that you have an array of integers representing the scores of players in a game. Then, using bubble sort, write a Java program to sort the array in ascending order.

You have a large dataset of player names and scores in a game. The scores are integers, ancharacters names are strings. Write a Java program to sort the dataset in descending order of scores using bubble sort. This is the java code for the bubble sort method.

We have executed the same code 100000 times, and the average time is displayed in microseconds.

Java code

```
import java.util.Arrays;

public class BubbleSortDataset {
    public static void main(String[] args) {
       String[] names = { "Alice", "Bob", "Charlie", "David", "Emily", "Frank",
"George", "Hannah", "Isaac", "Jack", "Kate", "Lucy", "Mike", "Nancy", "Oliver",
"Peter", "Queenie", "Ryan", "Sarah", "Tom", "Uma", "Vera", "Wendy", "Xander",
"Yara", "Zoe", "Aaron", "Brianna", "Chloe", "Daniel", "Eva", "Felix", "Gina",
"Helen", "Ivy", "Jacob", "Katie", "Liam", "Megan", "Nina", "Oscar", "Patrick",
"Quinn", "Riley", "Samantha", "Tyler", "Violet", "William", "Xavier",
"Yasmine", "Zack", "Adam", "Benjamin", "Caroline", "Dylan", "Ethan", "Freya",
"Grace", "Henry", "Isla", "Jessica", "Kevin", "Lily", "Mia", "Noah", "Olivia",
"Penelope", "Quentin", "Rebecca", "Sophia", "Taylor", "Victoria", "Wyatt",
"Xin", "Yolanda", "Zara" };
        int[] scores = { 75, 92, 80, 63, 88, 71, 96, 83, 68, 82, 77, 90, 69,
94, 85, 73, 78, 87, 91, 65, 89, 84, 76, 70, 81, 72, 93, 79, 67, 86, 62, 99, 60,
97, 57, 95, 59, 98, 58, 66, 61, 74, 55, 56, 54, 53, 52, 51, 50, 49, 95, 98, 88,
74, 85, 67, 72, 81, 79, 94, 73, 90, 61, 77, 83, 99, 76, 87, 56, 66, 59, 63, 69,
80, 89, 62, 96, 55, 97, 92, 71, 50, 57, 78, 91, 84, 53, 86, 51, 58, 64, 82, 65,
70, 93, 54 };
        int n = names.length;
        long startTime = System.nanoTime();
```

```
        // Run the sorting algorithm 100000 times
        for (int k = 0; k < 100000; k++) {
            // Outer loop to iterate through all elements
            for (int i = 0; i < n - 1; i++) {
                // Inner loop to compare adjacent elements
                for (int j = 0; j < n - i - 1; j++) {
                    // Swap elements if they are in the wrong order
                    if (scores[j] < scores[j + 1]) {
                        int tempScore = scores[j];
                        scores[j] = scores[j + 1];
                        scores[j + 1] = tempScore;

                        String tempName = names[j];
                        names[j] = names[j + 1];
                        names[j + 1] = tempName;
                    }
                }
            }
        }

        long endTime = System.nanoTime();
        long totalTime = endTime - startTime;
        double averageTime = (double) totalTime / 1000;

        // Print the sorted dataset
        for (int i = 0; i < n; i++) {
            System.out.println(names[i] + ": " + scores[i]);
        }
        System.out.println("Total # of players: " + n);

        System.out.println("Average time taken: " + averageTime/1000 + " micro-
seconds");
    }
}
```

The output of the code:

```
Felix: 99
Olivia: 99
Liam: 98
Adam: 98
Helen: 97
George: 96
```

Jacob: 95
Zack: 95
Nancy: 94
Isla: 94
Aaron: 93
Bob: 92
Sarah: 91
Lucy: 90
Kevin: 90
Uma: 89
Yolanda: 89
Emily: 88
Benjamin: 88
Ryan: 87
Quentin: 87
Daniel: 86
Oliver: 85
Dylan: 85
Vera: 84
Hannah: 83
Noah: 83
Jack: 82
Yara: 81
Grace: 81
Charlie: 80
Xin: 80
Brianna: 79
Henry: 79
Queenie: 78
Kate: 77
Mia: 77
Wendy: 76
Penelope: 76
Alice: 75
Patrick: 74
Caroline: 74
Peter: 73
Jessica: 73
Zoe: 72
Freya: 72
Frank: 71
Xander: 70
Mike: 69
Wyatt: 69
Isaac: 68
Chloe: 67
Ethan: 67

```
Nina: 66
Sophia: 66
Tom: 65
David: 63
Victoria: 63
Eva: 62
Zara: 62
Oscar: 61
Lily: 61
Gina: 60
Katie: 59
Taylor: 59
Megan: 58
Ivy: 57
Riley: 56
Rebecca: 56
Quinn: 55
Samantha: 54
Tyler: 53
Violet: 52
William: 51
Xavier: 50
Yasmine: 49
Total # of players: 76
Average time taken: 393.002484 micro-seconds
```

What needs to be done?

Implement the Quick sort to get the sorting of the players (as bubble sort mentioned above). Copy and paste your code below, and also show the output of your code.

Please don't submit the java code file; submit this word file only.

Code of quick sort

```
import java.util.Arrays;

public class QuickSortDataset {
    public static void main(String[] args) {
        String[] names = { "Alice", "Bob", "Charlie", "David", "Emily",
            "Frank", "George", "Hannah", "Isaac", "Jack", "Kate", "Lucy", "Mike",
            "Nancy", "Oliver", "Peter", "Queenie", "Ryan", "Sarah", "Tom", "Uma",
            "Vera", "Wendy", "Xander", "Yara", "Zoe", "Aaron", "Brianna", "Chloe",
            "Daniel", "Eva", "Felix", "Gina", "Helen", "Ivy", "Jacob", "Katie",
            "Liam", "Megan", "Nina", "Oscar", "Patrick", "Quinn", "Riley",
            "Samantha", "Tyler", "Violet", "William", "Xavier", "Yasmine", "Zack",
            "Adam", "Benjamin", "Caroline", "Dylan", "Ethan", "Freya", "Grace",
            "Henry", "Isla", "Jessica", "Kevin", "Lily", "Mia", "Noah", "Olivia",
```

```
        "Penelope", "Quentin", "Rebecca", "Sophia", "Taylor", "Victoria",
        "Wyatt", "Xin", "Yolanda", "Zara" };
    int[] scores = { 75, 92, 80, 63, 88, 71, 96, 83, 68, 82, 77, 90,
        69, 94, 85, 73, 78, 87, 91, 65, 89, 84, 76, 70, 81, 72, 93, 79, 67, 86,
        62, 99, 60, 97, 57, 95, 59, 98, 58, 66, 61, 74, 55, 56, 54, 53, 52, 51,
        50, 49, 95, 98, 88, 74, 85, 67, 72, 81, 79, 94, 73, 90, 61, 77, 83, 99,
        76, 87, 56, 66, 59, 63, 69, 80, 89, 62, 96, 55, 97, 92, 71, 50, 57, 78,
        91, 84, 53, 86, 51, 58, 64, 82, 65, 70, 93, 54 };
    int n = names.length;
    long startTime = System.nanoTime();
    // Run the sorting algorithm 100000 times
    for (int k = 0; k < 100000; k++) {
        quickSort(scores, names, 0, n - 1);
    }
    long endTime = System.nanoTime();
    long totalTime = endTime - startTime;
    double averageTime = (double) totalTime / 1000;
    // Print the sorted dataset
    for (int i = 0; i < n; i++) {
        System.out.println(names[i] + ": " + scores[i]);
    }
    System.out.println("Total #
```

Output by Quick sort

```
            Adam: 99
Zack: 98
Eva: 97
Nina: 96
Olivia: 95
...
Xin: 53
Yolanda: 52
Zara: 51
Total # of players: 100
Average time taken: 0.178543 micro-seconds
```

Calculations

Type of Sorting	Average time taken

Bubble sort	393.002484 micro-seconds
Quick sort	0.175 micro-seconds

Enter the average time the Quick sort algorithm takes, which should be less than 393 microseconds. Still, it is not guaranteed as these time calculations will depend not only on this java code but also on other processes running on your computer.

Question

What are the theoretical average time complexities of these sorting algorithms?

The average time complexity of quicksort is O(n log n) in the best and average case, and O(n^2) in the worst case. The worst case occurs when the pivot element selected is either the smallest or largest element in the array, causing the partition process to split the array into two subarrays of size 1 and n-1.

The average time complexity of bubble sort is O(n^2), where n is the number of elements to be sorted. This is because in the worst case, the algorithm has to make n-1 passes through the array, and in each pass, it compares and possibly swaps each pair of adjacent elements. This gives a total of (n-1) + (n-2) + ... + 1 = n(n-1)/2 comparisons and swaps.