

Homework #1

In this assignment you will implement a program that inserts elements into an array, analyze the Big-O performance of this code, then profile the program to see if the actual performance matches the predicted performance.

All code implemented in this assignment should be in a class called `Homework1`.

1. **(2 points)** Implement a method named `insert`. This method should take an array of `ints`, the index at which a new value should be inserted, and the new value that should be inserted. The function should return a new array populated with the contents of the original array with the given value inserted at the given index. The following sections provide a detailed description of this function:

Method signature:

```
static int[] insert(int[] array, int index, int value)
```

Parameters:

<code>array</code>	The original array of <code>ints</code> .
<code>index</code>	The location where the value will be inserted.
<code>value</code>	The value to be inserted.

Return value:

A new array of `ints` containing the contents of the original array plus the new value inserted at the given index.

Pseudocode:

```
// Create new array one larger than original array
Let newArray = a newArray with array.length + 1 elements

// Copy elements up to insert point from original array to new array
Loop to copy array[0, index) to newArray[0, index)

// Place insert value into new array
Set newArray[index] to value

// Copy elements after insert point from original array to new array
Loop to copy array[index, length) to newArray[index + 1, length + 1)

Return newArray
```

2. **(2 points)** Implement a main method that profiles the performance of `insert` and outputs a table showing the average time per insert as the length of the array increases.

Pseudocode:

```
// Setting to allow fine-tuning the granularity of the readings
Let NUM_READINGS = 60
Let INSERTS_PER_READING = 1000

// Start with an array containing 1 element
Let array = new array containing one element having value 0

// Take NUM_READINGS readings
Loop NUM_READINGS times
    // Each reading will be taken after INSERTS_PER_READING inserts
    Let startTime = current time
    Loop INSERTS_PER_READING times
        Let index = random integer in range [0, array.length)
        Let value = random integer value
        Let array = Homework1.insert(array, index, value)
    End Loop
    Let stopTime = current time
    Let timePerInsert = (stopTime - startTime) / INSERTS_PER_READING

    // Output reading in tabular format
    Output array length and timePerInsert
End Loop
```

Report format:

Your program should output a report similar to the format below (your values will be different). You should fine-tune the `INSERTS_PER_READING` constant so that none of the readings ("Seconds per insert") are zero:

Array length	Seconds per insert
1000	0.000024
2000	0.000028
3000	0.000041
4000	0.000036
...	...
57000	0.000262
58000	0.000318
59000	0.000324
60000	0.000328

3. **(2 points)** Plot a scatter graph showing “Seconds per insert” (Y-axis) vs. “Array length” (X-axis) using the profiling data that was output by main.
4. **(2 points)** Provide a line-by-line Big-O analysis of your implementation of insert. You can do this by adding a comment next to each line in your source code. What is the overall Big-O performance of insert? What parts of the algorithm contribute most heavily to the overall Big-O performance?
5. **(1 point)** Based on the graph does the performance of improve, degrade, or stay the same as the length of the array grows? Does your Big-O analysis of match the results of running the program?
6. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, and follows programming best-practices.

Turn in all source code, program output, diagrams, and answers to questions in a single PDF document.