**Program Structure and Algorithms (INFO-6205 SEC01)**

**Assignment – 2**

**Benchmark**

Name: Vivek Sharma

NUID: 002105272

**TASK**

Task for this assignment is in three parts.

1. Implement three methods (*repeat*, *getClock*, and *toMillisecs*) of a class called *Timer*.
2. Implement Insertion Sort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort.
3. Implement a main program to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially ordered and reverse ordered.

**OUTPUT SCREENSHOT**

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated

**CONCLUSION**

Reverse-ordered array has the highest sorting time according to the benchmark. It is followed by random-ordered arrays, then partial-ordered arrays and then ordered arrays. The worst-case scenario is for reverse ordered arrays sorting as it has O(N2) time complexity.

**EVIDENCE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Random | Ordered | Reverse | Partial\_ordered |
| 50 | 1.7 | 0 | 0.3 | 0 |
| 100 | 0.2 | 0 | 0.2 | 0 |
| 200 | 0.1 | 0 | 0.3 | 0 |
| 400 | 0.1 | 0 | 0.3 | 0 |
| 800 | 0.5 | 0 | 1 | 0.3 |
| 1600 | 2.2 | 0 | 4.3 | 1 |
| 3200 | 8.8 | 0 | 19.2 | 4.1 |

**UNIT TESTS:**

**Graphical user interface, text

Description automatically generated**

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated