**Practical 1**

**Question 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sorting Method** | **Array Size** | **Sorting Option** | **Results** |
| Bubble Sort | 50000 | Ascending | 1129.0 |
|  | 10000 | Descending | 178702.33 |
|  | 1000 | Random | 2194.33 |
|  | 100 | Nearly Sorted | 113.0 |
| Selection Sort | 50000 | Ascending | 1401365.33 |
|  | 10000 | Descending | 56496.66 |
|  | 1000 | Random | 3369.0 |
|  | 100 | Nearly Sorted | 228.0 |
| Insertion Sort | 50000 | Ascending | 1573.33 |
|  | 10000 | Descending | 23400.667 |
|  | 1000 | Random | 31478.0 |
|  | 100 | Nearly Sorted | 15.0 |

Bubble Sort was the fastest for sorting in ascending. Insertion Sort as the fastest for descending and nearly sorted. Selection was the fastest for random. In terms of handling larger array sizes, Bubble sort still showed superiority over other sorting methods. Taking into account the time complexity of each sorting algorithm, bubble sort is not the most effective due to the sheer amount of swaps needed. Bubble is effective strictly when the data is already sorted and is also easy to implement. Insertion sort was quite fast relative to other algorithms throughout all types of data, however, slightly struggles with reversed data. Insertion sort is also more complex to implement. Finally, selection sort was also quite fast as it does minimal work per pass. The downside of this is that regardless of how sorted the data is already, selection sort takes no advantage of this and still takes the same amount of steps. Selection sort also sorts unstably whilst the other two algorithms have stable sorts.