**Practical 02**

***Sorting Algorithms – Selection Sort and Insertion Sort***

Selection Sort and Insertion Sort are quadratic sorting algorithms that operate in the order of O(n2), The following practical will help you grasp how these algorithms are implemented and how the abstract concept of these sorting algorithms can differ from their concrete implementations.

**Consider the following array of Integer values.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **22** | **11** | **34** | **-5** | **3** | **40** | **9** | **16** | **6** |

1. Write the contents of the array for 5 passes of the outermost loop of insertion sort.

**Pass 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 3**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 4**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 5**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

1. Write the contents of the array for 4 passes of the outermost loop of selection sort.

**Pass 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 3**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Pass 4**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

1. Create a Selection Sort class that has a sorting method that will accept an array as an argument and will return a sorted array.
2. Create an Insertion Sort class that has a sorting method that will accept an array as an argument and will return a sorted array.
3. Write a piece of code that will propagate a static array of 10000 random elements.
4. Write a piece of code that will propagate a static array of 10000 sorted elements.
5. Write a piece of code that will propagate a static array of 10000 inversely sorted (sorted z – a).
6. Implement an array list, populating it similar to the arrays in 5, 6 and 7 above.
7. Rewrite the Insertion Sort and Selection Sort algorithms to sort the array lists.
8. Does changing the data structure from a static array to an array list have an impact on the overall efficiency of the algorithm?
9. Create a print method that will print both the initial array and the sorted version.
10. Write a print method that will validate your results in steps 1 and 2 above.
11. Write code that will record the number of comparisons that the Selection and Insertion Sort algorithms will carry out on the datasets implemented in steps 5, 6, and 7 above.
12. Write code that will record the number of swap operations that the Selection and Insertion Sort algorithms will carry out on the datasets implemented in steps 5, 6, and 7 above.