CS346 :- SOFTWARE ENGINEERING LAB
ASSIGNMENT 1 - MILESTONE 1
19th January 2024

Exploring & Visualising Insertion Sort

A COMPREHENSIVE REPORT

This report covers the brief implementation of the software to visualise the insertion sort algorithm on an array of integers.

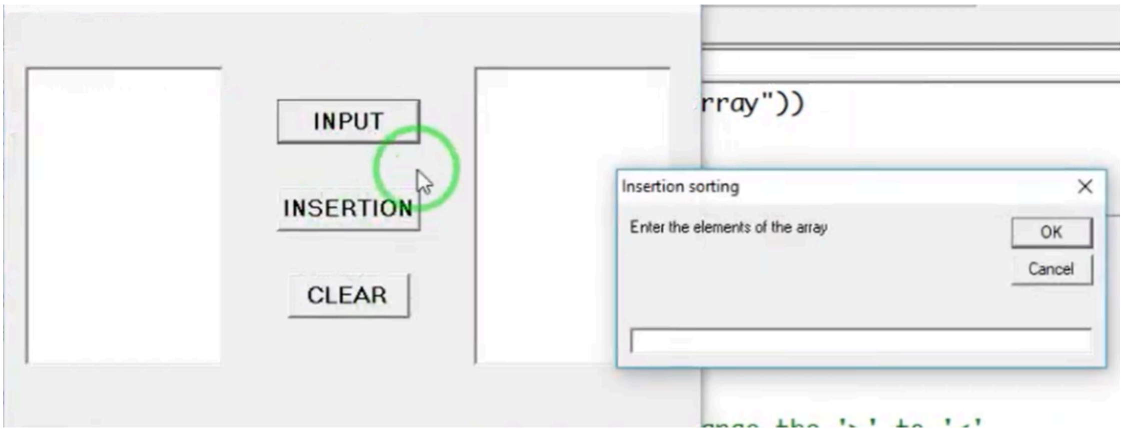
NAME : - VATSAL JAIN
DEPT :- CSE (B.TECH 3rd YEAR)
ROLL NO :- 210101110

Problem Statement

We have to implement a software which sorts an array of integers using insertion sort algorithm. We have to include necessary checks and the UI graphics to visualise insertion sort in a much easier way.

- **Input Format :-** We take an integer N (number of elements in the array), as input from the user. Then we take the input of n integers, separated by spaces.
- **Output Format :-** We display the actual array after each pass of the insertion sort, as well as the completely sorted array at the end of the insertion sort algorithm.

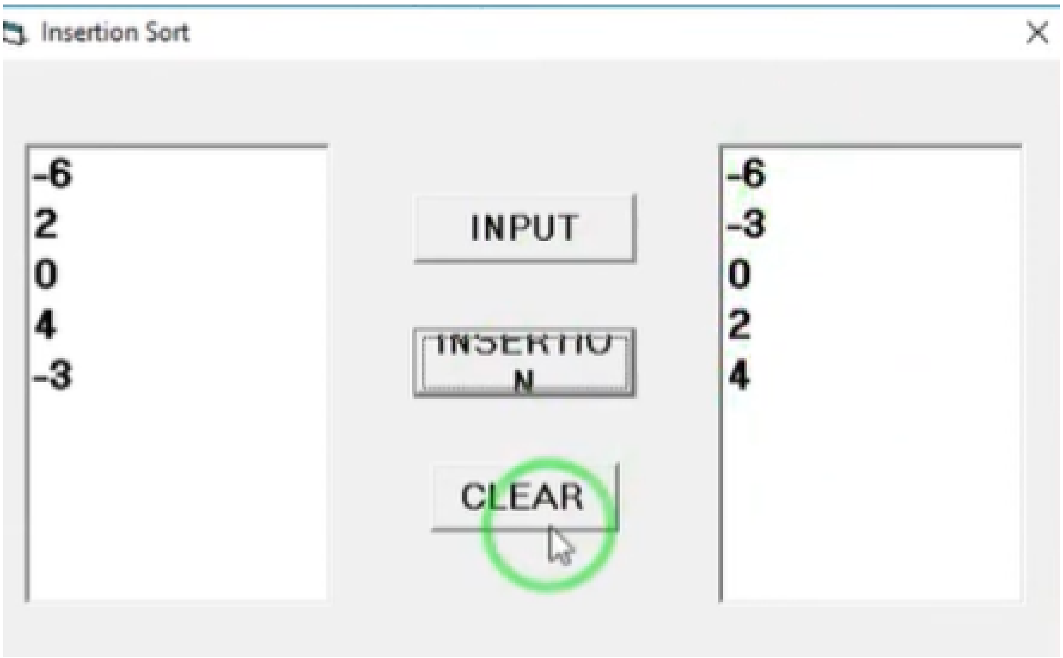
Here are the images depicting the sample UI of the software :-



rray"))

Insertion sorting

Enter the elements of the array



Insertion Sort

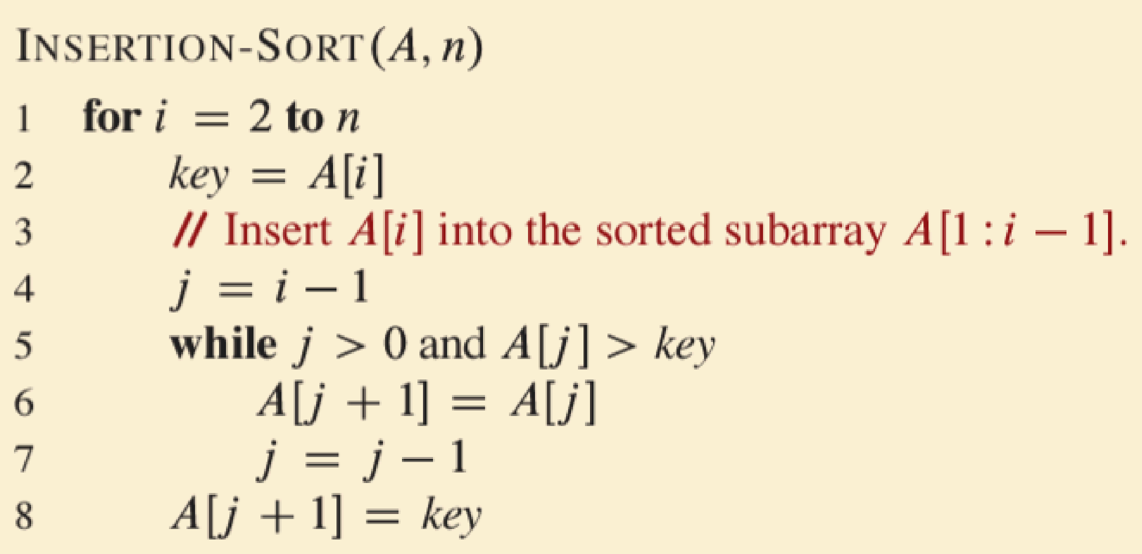
```
-6
2
0
4
-3
```

```
-6
-3
0
2
4
```

Insertion Sort Algorithm

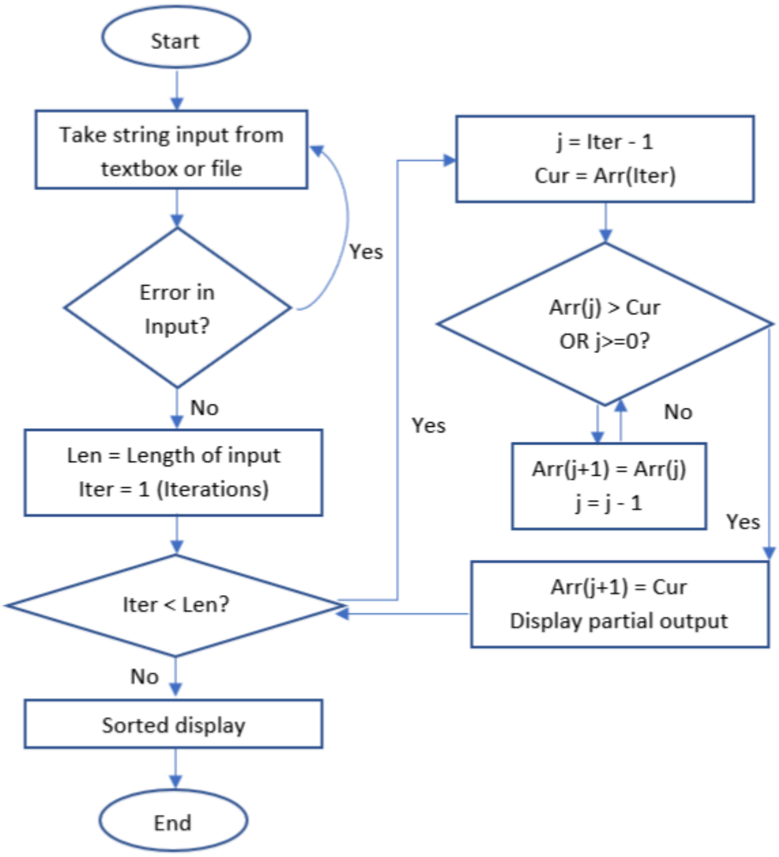
The Insertion Sort algorithm has a simple idea. Assume an array with items to be sorted. We divide the array into two parts: sorted one and unsorted one. At the beginning, the sorted part consists of the first element. Then, for each item that we have in the unsorted part,we take the element and insert it into the right place among the sorted items.

The **pseudo code** for the insertion sort algorithm is as follows :-



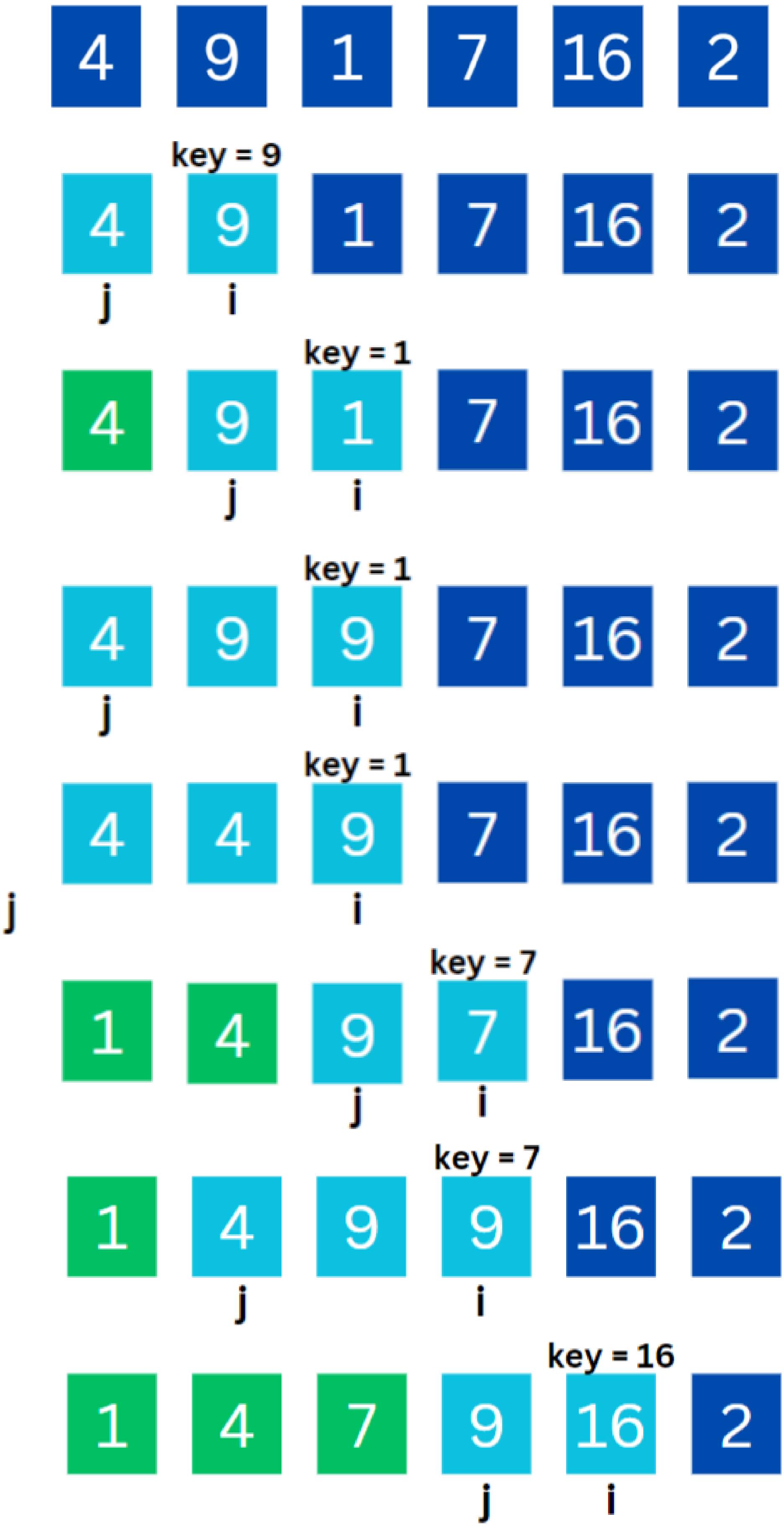
```
INSERTION-SORT(A, n)
for $i = 2$ to $n$
    $key = A[i]$
    // Insert $A[i]$ into the sorted subarray $A[1 : i - 1]$.
    $j = i - 1$
    while $j > 0$ and $A[j] > key$
        $A[j + 1] = A[j]$
        $j = j - 1$
    $A[j + 1] = key$
```

The **flowchart** depicting the working of the insertion sort algorithm is as follows :-



Insertion Sort Algorithm on an Array of Integers

Here is an example of how the insertion sort algorithm actually works :-



this is the
initial array.

arr[j] <= key, so while
loop terminates,
arr[j+1] = key

arr[j]>key, so as per
the while loop,
arr[j+1] = arr[j]

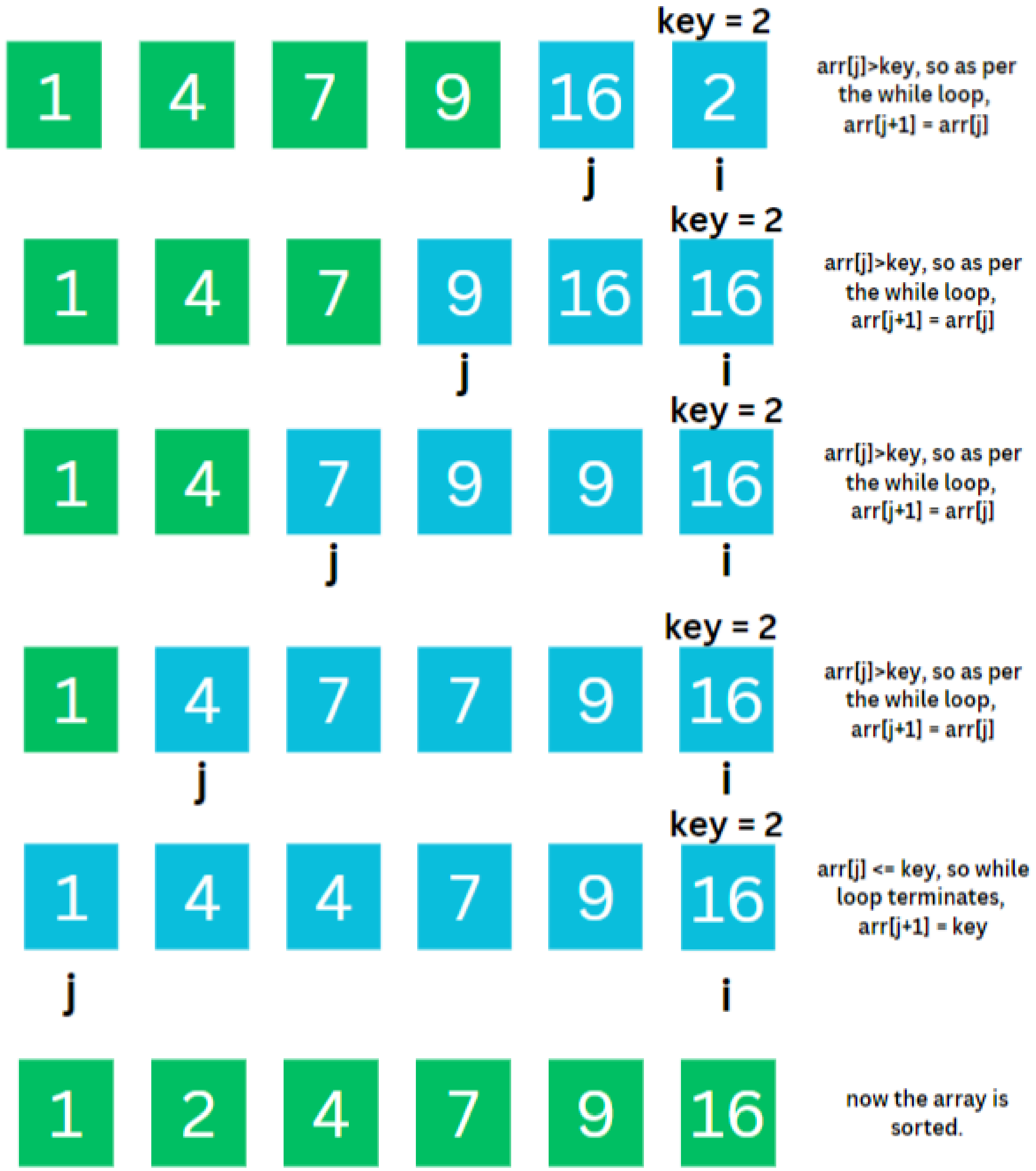
arr[j]>key, so as per
the while loop,
arr[j+1] = arr[j]

j<0, so while loop
terminates,
arr[j+1] = key

arr[j]>key, so as per
the while loop,
arr[j+1] = arr[j]

arr[j] <= key, so while
loop terminates,
arr[j+1] = key

arr[j] <= key, so while
loop terminates,
arr[j+1] = key



arr[j]>key, so as per the while loop, arr[j+1] = arr[j]

arr[j]>key, so as per the while loop, arr[j+1] = arr[j]

arr[j]>key, so as per the while loop, arr[j+1] = arr[j]

arr[j]>key, so as per the while loop, arr[j+1] = arr[j]

arr[j] <= key, so while loop terminates, arr[j+1] = key

now the array is sorted.

Complexity Analysis of Insertion Sort

- **Time Complexity :-** Insertion sort takes the maximum time to sort if elements are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted. The best case time complexity of insertion sort is O(n), whereas the worst and average case time complexity is O(n^2) for both.
- **Space complexity :-** The auxiliary space complexity of Insertion Sort is O(1).

Design Goals & Input/Output Restrictions

- The input consists of both the number of elements and the elements within the array. The software is responsible for validating this input by implementing the necessary checks.
- The program should output the array after each pass of the for loop and the final sorted array, using the insertion sort algorithm implemented.
- The number of elements and the elements in the array should strictly be integers, which would be ensured using the necessary checks

Conclusion

The aim of this report was to give a very basic idea about the working of insertion sort and the overall design of the software to be implemented. The in-depth analysis of the working software would be covered in the subsequent parts of the assignment.