LAB211 Assignment

Type:	**Short Assignment**
Code:	**J1.S.P0003**
LOC:	**40**
Slot(s):	**1**

Title

Insertion sort algorithm.

Background Context

Insertion sort belongs to the $O(n^2)$ sorting algorithms. Unlike many sorting algorithms with quadratic complexity, it is actually applied in practice for sorting small arrays of data. For instance, it is used to improve quicksort routine. Some sources notice, that people use same algorithm ordering items, for example, hand of cards.

Program Specifications

Design a program that allows users to input the number of array. Generate random integer in number range input. Display unsorted array and sorted array using insertion sort.

Function details:

1. Display a screen to prompt users to input a positive decimal number.
   - Users run the program, display a screen to ask users to enter a positive decimal number.
   - Users input a positive decimal number. Then, perform **Function 2**.
2. Display & sort array.
   - Generate random integer in number range for each array element.
   - Display array before and after sorting.

Expectation of User interface:

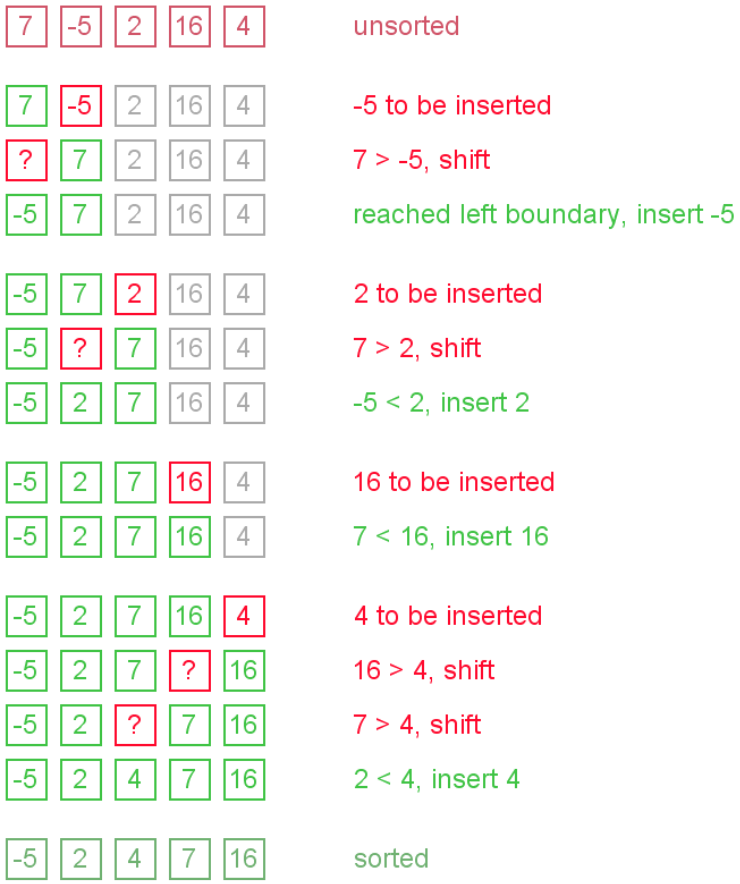
```
Enter number of array:
10
Unsorted array: [2, 6, 3, 6, 8, 6, 1, 2, 9, 8]
Sorted array: [1, 2, 2, 3, 6, 6, 6, 8, 8, 9]BUILD SUCCESSFUL (total time: 1 second)
```

Guidelines

Algorithm

Insertion sort algorithm somewhat resembles selection sort. Array is imaginary divided into two parts - **sorted one** and **unsorted one**. At the beginning, **sorted part** contains **first element** of the array and **unsorted one** contains the rest. At every step, algorithm takes **first element** in the **unsorted part** and **inserts** it to the right place of the **sorted one.** When **unsorted part** becomes **empty**, algorithm *stops*. Sketchy, insertion sort algorithm step looks like this:

The idea of the sketch was originally posted http://en.wikipedia.org/wiki/Insertion_sort.
Let us see an example of insertion sort routine to make the idea of algorithm clearer.
Example. Sort {7, -5, 2, 16, 4} using insertion sort.



The ideas of insertion

The main operation of the algorithm is **insertion**. The task is to insert a value into the sorted part of the array. Let us see the variants of how we can do it.

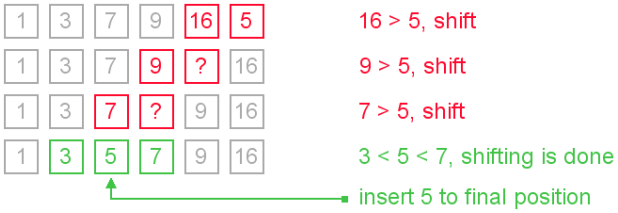
"Sifting down" using swaps

The simplest way to insert next element into the sorted part is to sift it down, until it occupies correct position. Initially the element stays right after the sorted part. At each step algorithm compares the element with one before it and, if they stay in reversed order, swap them. Let us see an illustration.

This approach writes sifted element to temporary position many times. Next implementation eliminates those unnecessary writes.

Shifting instead of swapping

We can modify previous algorithm, so it will write sifted element only to the final correct position. Let us see an illustration.



It is the most commonly used modification of the insertion sort.