**Assignment Number**

**Problem Statement**

Program in C to implement Insertion sort in ascending order.

**Theory**

**Insertion sort** is a simple [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm) that builds the final [sorted array](https://en.wikipedia.org/wiki/Sorted_array) (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as [quicksort](https://en.wikipedia.org/wiki/Quicksort), [heapsort](https://en.wikipedia.org/wiki/Heapsort), or [merge sort](https://en.wikipedia.org/wiki/Merge_sort). However, insertion sort provides several advantages:

* Efficient for (quite) small data sets, much like other quadratic sorting algorithms
* More efficient in practice than most other simple quadratic (i.e., [O](https://en.wikipedia.org/wiki/Big_O_notation)(*n*2)) algorithms such as [selection sort](https://en.wikipedia.org/wiki/Selection_sort) or [bubble sort](https://en.wikipedia.org/wiki/Bubble_sort)
* [Adaptive](https://en.wikipedia.org/wiki/Adaptive_sort), i.e., efficient for data sets that are already substantially sorted: the [time complexity](https://en.wikipedia.org/wiki/Time_complexity) is *O*(*nk*) when each element in the input is no more than *k* places away from its sorted position
* [Stable](https://en.wikipedia.org/wiki/Stable_sort); i.e., does not change the relative order of elements with equal keys
* [In-place](https://en.wikipedia.org/wiki/In-place_algorithm); i.e., only requires a constant amount O(1) of additional memory space
* [Online](https://en.wikipedia.org/wiki/Online_algorithm); i.e., can sort a list as it receives it

**Complexity:**

* Best case: O(n)
* Worst case: O(n2)
* Average case: O(n2)

**Algorithm**

**Input :** An unsorted array, say **a[].**

**Output :** Elements of the input array **a[]** sorted in ascending order**.**

**Steps :**

1. Print "Enter the number of elements of the array: "
2. Input n
3. Repeat Step 3.a to Step 3.b For i=0 to i<n
   1. Print "Enter the element no. "i+1
   2. Input a[i]
4. Print "The sorted array is: "
5. Repeat Step 5.a to Step 5.7 For i=1 to i<n
   1. Set key=a[i]
   2. Set j=i-1
   3. Repeat Step 5.c.i while(j>=0 AND a[j]>key)
      1. Set a[j+1]=a[j-1]
      2. Set j = j - 1
6. a[j+1]=key
7. Set i=i+1
8. Repeat Step 6.i to Step 6.ii For i=0 to i<n
   * 1. Print a[i]
     2. Set i=i+1

**Source Code**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*a,i,j,min,t,n,k;

printf("Enter the number of elements of the array: ");

scanf("%d",&n);

a=(int\*)malloc(n\*sizeof(int));

for(i=0;i<n;i++){

printf("Enter the element no. %d: ",i+1);

scanf("%d",a+i);

}

printf("The sorted array is: \n");

for (i=1;i<n;i++){

key=a[i];

j=i-1;

/\* Move elements of a[0..i-1], that are

greater than key, to one position ahead

of their current position \*/

while(j>=0&&a[j]>key){

a[j+1]=a[j--];

}

a[j+1]=key;

}

for(i=0;i<n;i++){

printf("%d\n",\*(a+i));

}

return 0;

}

**Input and Output**

**Set 1 :**

Enter the number of elements of the array: 5

Enter the element no. 1: -381

Enter the element no. 2: 382

Enter the element no. 3: 481

Enter the element no. 4: 0

Enter the element no. 5: 38

The sorted array is:

-381

0

38

382

481

**Set 2 :**

Enter the number of elements of the array: 5

Enter the element no. 1: 382

Enter the element no. 2: 39

Enter the element no. 3: 981

Enter the element no. 4: 28

Enter the element no. 5: 30

The sorted array is:

28

30

39

382

981

**Discussion**

1. Insertion sort is a[daptive](https://en.wikipedia.org/wiki/Adaptive_sort), i.e., efficient for data sets that are already substantially sorted: the [time complexity](https://en.wikipedia.org/wiki/Time_complexity) is *O*(*nk*) when each element in the input is no more than *k* places away from its sorted position
2. Insertion sort is s[table](https://en.wikipedia.org/wiki/Stable_sort); i.e., does not change the relative order of elements with equal keys
3. Insertion sort can be done i[n-place](https://en.wikipedia.org/wiki/In-place_algorithm); i.e., requiring no additional memory space
4. Insertion sort is o[nline](https://en.wikipedia.org/wiki/Online_algorithm); i.e., can sort a list as it receives it