ADVANCED ALGORITHMS LAB

ASSIGNMENT - 1

INSERTION SORT

Name - AMAR PARAMESWARAN

Class - TYCS A

PRN - 19070122016

Advanced Algorithm Lab Assignment No:

Name of the Student	PRN	Batch	TY CS/IT Division
Amar Parameswaran	19070122016	TY CS	A

TOPIC: INSERTION SORT USING C/C++ (RECURSIVE AND ITERATIVE)

INSERTION SORT

WORKING MECHANISM:

It is a sorting mechanism where we sort the array by comparing elements in the array sequentially.

We basically follow one order it may be ascending or descending order depending on the question, and we can sort the elements by comparing them with each other and whether they must be placed to the left or right of any element.

EXAMPLE:

INPUT: 6 3 4 8

OUTPUT: 3 4 6 8

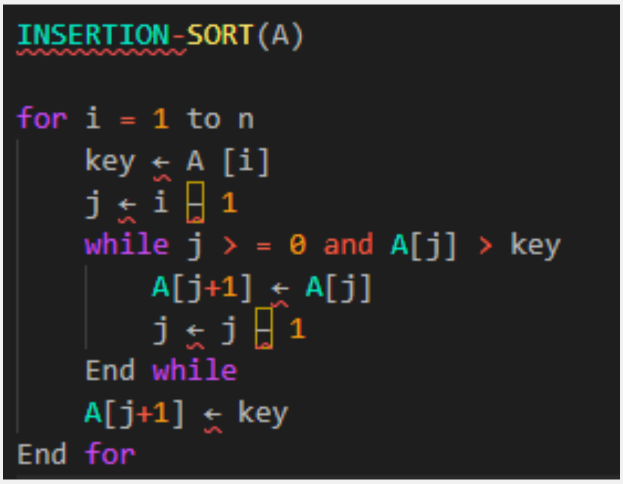
Here element 3 is compared with 6 and moved to the left of 6 - 3 6 4 8

Then 4 is moved as it is less than 6 - 3 4 6 8

As 8 is greater than 6 it remains where it is.

Final Output: 3 4 6 8

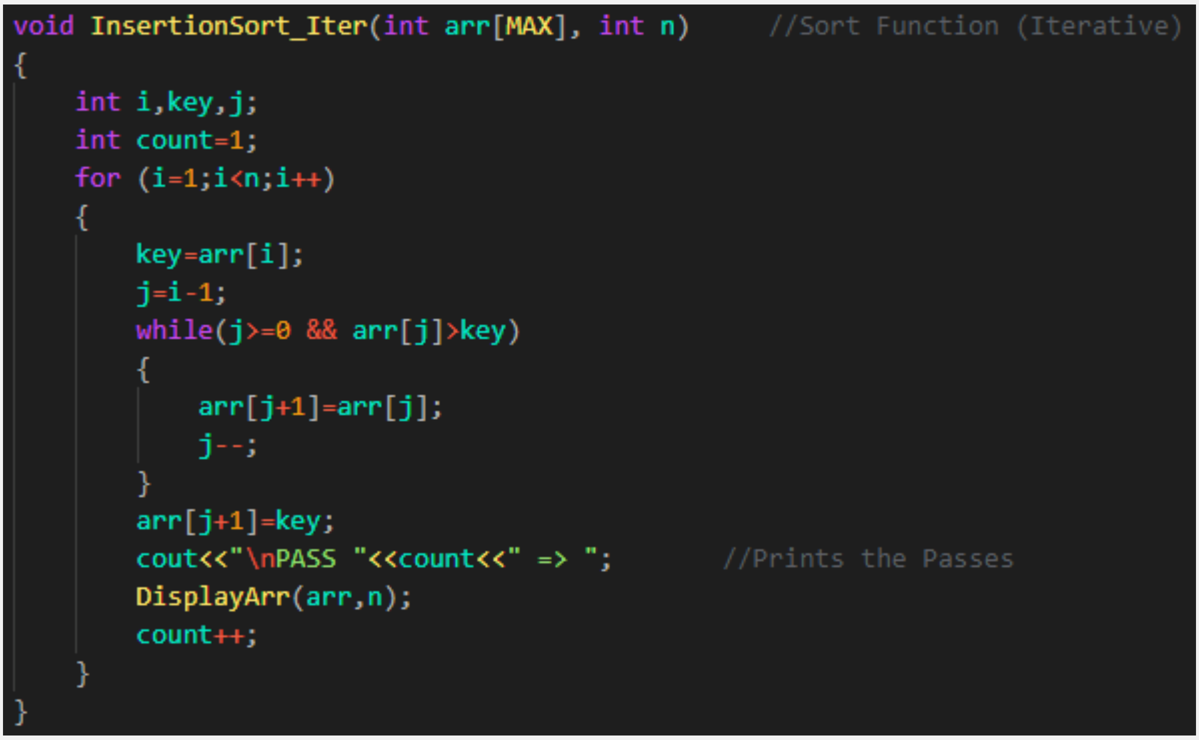
PSEUDO CODE:



```
INSERTION-SORT(A)

for i = 1 to n
    key ← A [i]
    j ← i ▯ 1
    while j > = 0 and A[j] > key
        A[j+1] ← A[j]
        j ← j ▯ 1
    End while
    A[j+1] ← key
End for
```

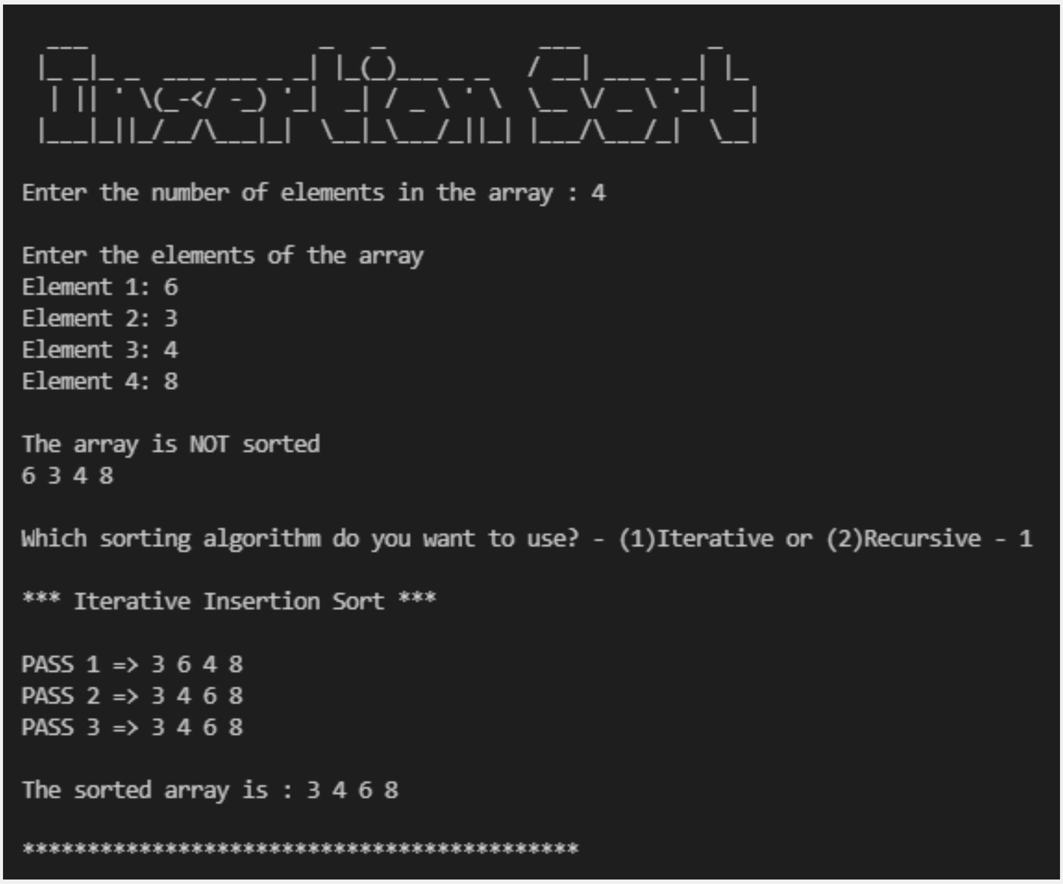
CODE SNIPPET:

ITERATIVE



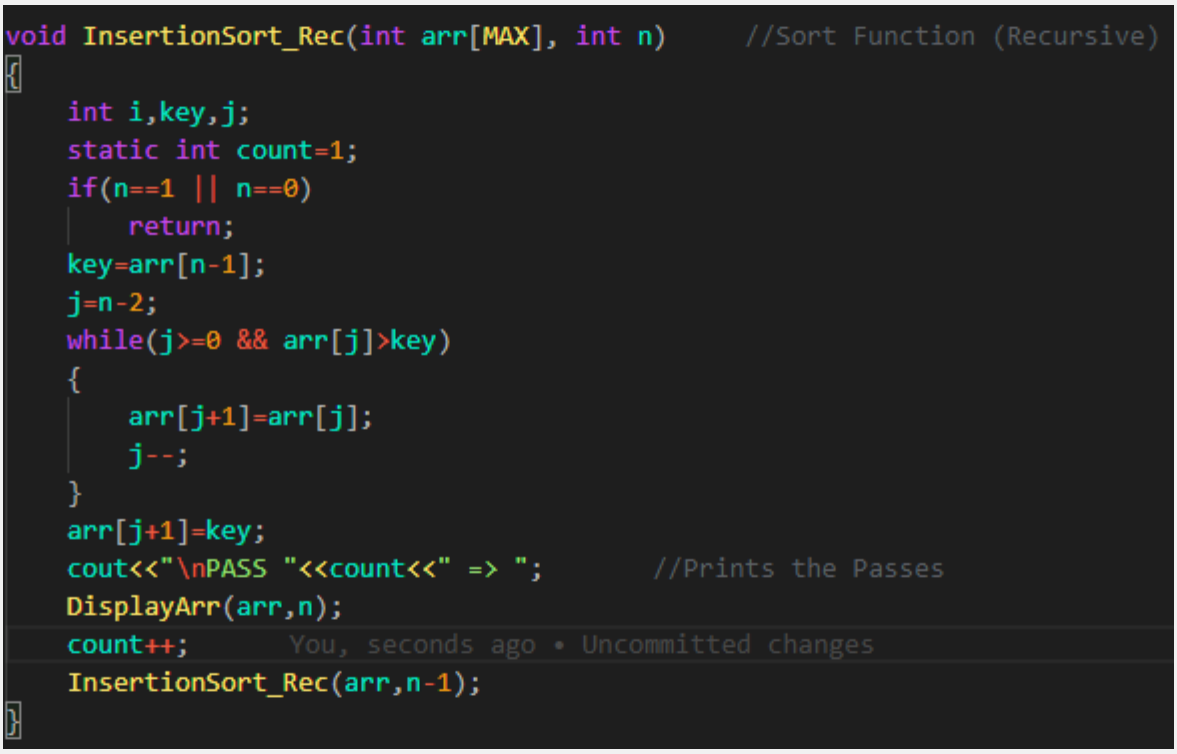
```
void InsertionSort_Iter(int arr[MAX], int n)      //Sort Function (Iterative)
{
    int i,key,j;
    int count=1;
    for (i=1;i<n;i++)
    {
        key=arr[i];
        j=i-1;
        while(j>=0 && arr[j]>key)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=key;
        cout<<"\nPASS "<<count<<" => ";       //Prints the Passes
        DisplayArr(arr,n);
        count++;
    }
}
```

OUTPUT



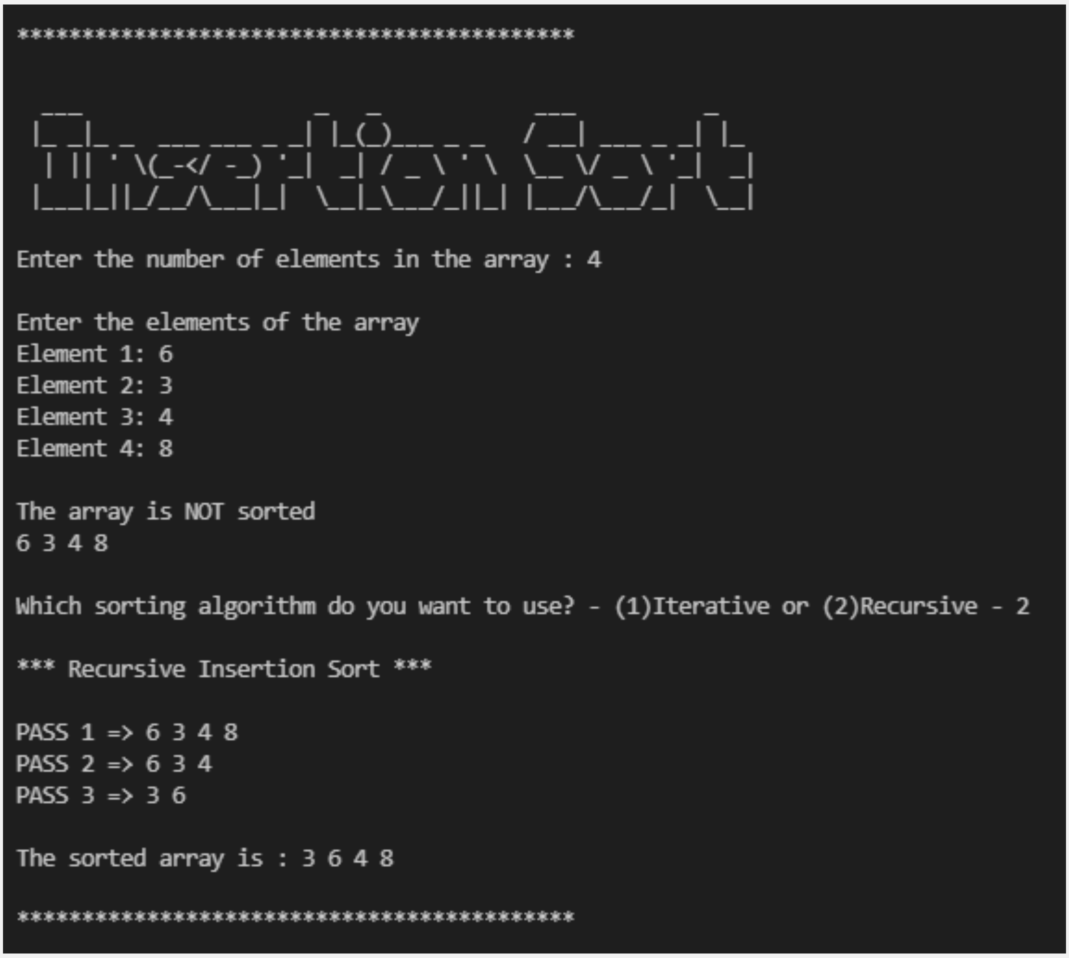
```
 ___                 _   _            ___          _
|_ _|_ _  ___ ___ _ _| |_(_)___ _ _   / __| ___ _ _| |_
 | || ' \(_-</ -_) '_|  _| / _ \ ' \  \__ \/ _ \ '_|  _|
|___|_||_/__/\___|_|  \__|_\___/_||_| |___/\___/_|  \__|

Enter the number of elements in the array : 4

Enter the elements of the array
Element 1: 6
Element 2: 3
Element 3: 4
Element 4: 8

The array is NOT sorted
6 3 4 8

Which sorting algorithm do you want to use? - (1)Iterative or (2)Recursive - 1

*** Iterative Insertion Sort ***

PASS 1 => 3 6 4 8
PASS 2 => 3 4 6 8
PASS 3 => 3 4 6 8

The sorted array is : 3 4 6 8

*******************************************
```

RECURSIVE



```
void InsertionSort_Rec(int arr[MAX], int n)     //Sort Function (Recursive)
{
    int i,key,j;
    static int count=1;
    if(n==1 || n==0)
        return;
    key=arr[n-1];
    j=n-2;
    while(j>=0 && arr[j]>key)
    {
        arr[j+1]=arr[j];
        j--;
    }
    arr[j+1]=key;
    cout<<"\nPASS "<<count<<" => ";        //Prints the Passes
    DisplayArr(arr,n);
    count++;        You, seconds ago • Uncommitted changes
    InsertionSort_Rec(arr,n-1);
}
```

OUTPUT



```
*******************************************


  ___                  _   _            ___          _
 |_ _|_ _  ___ ___ _ _| |_(_)___ _ _   / __| ___ _ _| |_
  | || ' \(_-</ -_) '_|  _| / _ \ ' \  \__ \/ _ \ '_|  _|
 |___|_||_/__/\___|_|  \__|_\___/_||_| |___/\___/_|  \__|

Enter the number of elements in the array : 4

Enter the elements of the array
Element 1: 6
Element 2: 3
Element 3: 4
Element 4: 8

The array is NOT sorted
6 3 4 8

Which sorting algorithm do you want to use? - (1)Iterative or (2)Recursive - 2

*** Recursive Insertion Sort ***

PASS 1 => 6 3 4 8
PASS 2 => 6 3 4
PASS 3 => 3 6

The sorted array is : 3 6 4 8

*******************************************
```

Amar Parameswaran 19070122016 TY CS A

Time complexity and space complexity

★ Time complexity

worst case : $O(N^2)$

Average case : $O(N^2)$

Best case : $O(N)$

Run time of insertion sort

Steps	Cost	Times
for $j=1$ to $n-1$	C_1	n
key = $A[j]$	C_2	$n-1$
$i=j-1$	C_3	$n-1$
while $i>0$ and $A[i] > key$	C_4	$\sum_{j=1}^{n-1} t_j$
do $A[i+1] = A[i]$	C_5	$\sum_{j=1}^{n-1} (t_j-1)$
$i = i-1$	C_6	$\sum_{j=1}^{n-1} (t_j-1)$
$A[i+1] = key$	C_7	$n-1$

Total running time of insertion sort

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=1}^{n-1} t_j + c_5 \sum_{j=1}^{n-1} (t_j - 1)$$
$$+ c_6 \sum_{j=1}^{n-1} (t_j - 1) + c_7(n-1)$$

$\Rrightarrow$ For Best Case Analysis

$t_j = 1$

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

Thus $O(n)$

$\Rrightarrow$ For worst Case Analysis

$t_j = j$

$$T(n) = \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right)n^2 + \left(c_1 + c_2 + c_3 - \frac{c_4}{2} - 3\frac{c_5}{2} - 3\frac{c_6}{2}\right)n$$
$$+ (c_5 + c_6 - c_2 - c_3 - c_7)$$

Thus $O(n^2)$

$\Rrightarrow$ For Average case Analysis

$t_j = j/2$ (assume)

$T(n)$ is still a quadratic (Put $T_j = j/2$ in $T(n)$)

Thus $O(n^2)$

★ Space complexity

There is no need for any extra space while performing insertion sort as only rearrangement of input occurs and no auxillary memory is needed.

→ Thus, memory/space complexity of insertion sort is $O(1)$

REAL LIFE APPLICATIONS

1. Shirts kept in a shop are in ascending order i.e., from small to large size which is an example of insertion sort
2. Playing cards

OPTIMIZATIONS

When the array has been entered in sorted order then insertion sort is in a state of optimization where it works best.

NOTE: If the dataset is large, it is better to use other sorting techniques like quick sort etc.