Class: SE_Computer

Subject: AOA RollNo_Name: 55_AdnanShaikh

EXPERIMENT NO- 2

AIM: Implementation and analysis of Insertion Sort.

PROBLEM STATEMENT :

WAP to sort given numbers using Insertion Sort algorithm.

Resource Required: Pentium IV, Turbo C, Printer, Printout Stationary

THEORY:

Insertion Sort: -

1. <u>Step 1:</u> The second element of an array is compared with the elements that appears before it (only first element in this case). If the second element is smaller than first element, second element is inserted in the position of first element. After first step, first two elements of an array will be sorted.
2. <u>Step 2:</u> The third element of an array is compared with the elements that appears before it (first and second element). If third element is smaller than first element, it is inserted in the position of first element. If third element is larger than first element but, smaller than second element, it is inserted in the position of second element. If third element is larger than both the elements, it is kept in the position as it is. After second step, first three elements of an array will be sorted.

3. <u>Step 3:</u> Similarly, the fourth element of an array is compared with the elements that appears before it (first, second and third element) and the same procedure is applied and that element is inserted in the proper position. After third step, first four elements of an array will be sorted.

If there are *n* elements to be sorted. Then, this procedure is repeated *n-1* times to get sorted list of array.

Algorithm:

Algorithm **insertion-sort** (an array A of n comparable objects)

```
START
(1)    for  p = 2 through n  do
(2)         temp = A[p];
(3)         for j = p though 2 do until  A[j-1]<temp
(4)              A[j] = A[j-1]; // shift all larger elements right

(5)         A[j-1] = temp; // insert it in its correct position in the
                            // already sorted portion

END
```

EXAMPLE:

34 **8** 64 51 32 21

8 34 **64** 51 32 21

8 34 *64* **51** 32 21

8 34 *51* 64 **32** 21

8 *32* 34 51 64 **21**

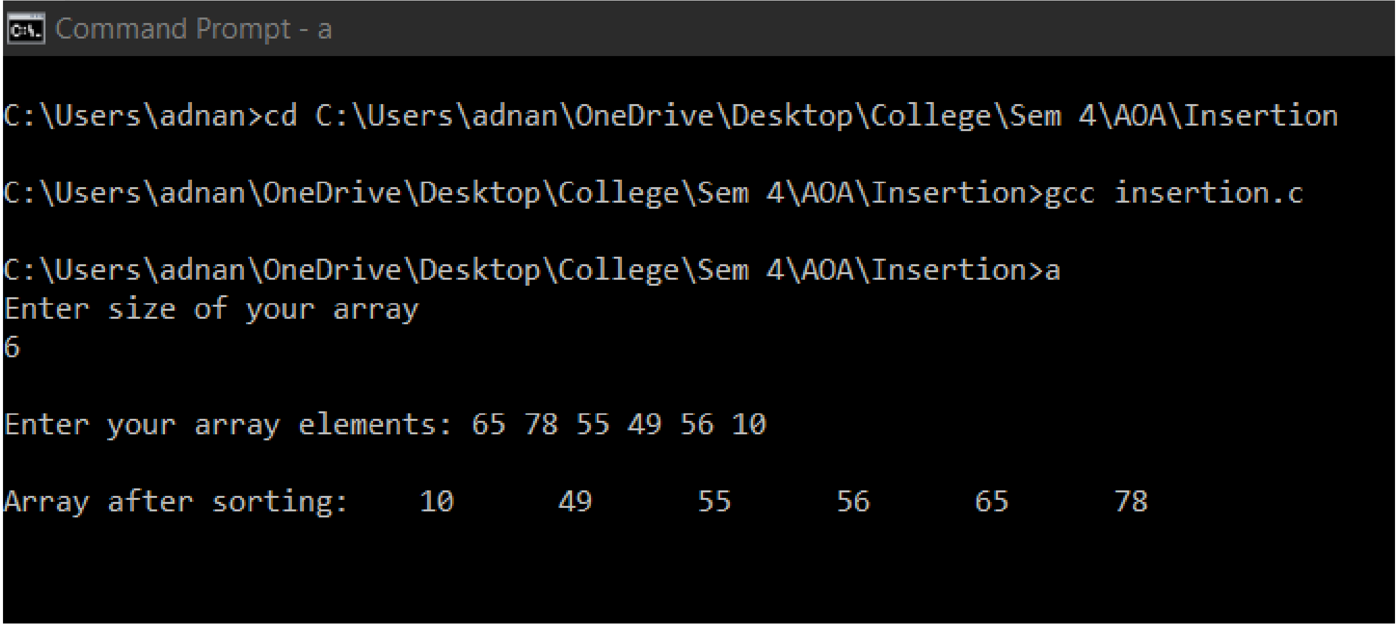
8 *21* 32 34 51 64.

CONCLUSION: In Insertion Sort if the initial array is sorted, only one comparison is made on each pass, so that Best case time complexity is O(n). Average and worst case time Complexity is O(n^2). As with selection sort, insertion sort is an in-place sort. Its asymptotic space complexity is O(n).

Code:

```
#include<stdio.h>
#include<conio.h>

void insert(int arr[],int z);

int main()
{
    int z;
    printf("Enter size of your array \n");
    scanf("%d",&z);
    int arr[z];
    printf("\nEnter your array elements: ");
    for (int i = 0;i<z;i++)
    {
        scanf("%d",&arr[i]);
    }
    insert(arr,z);
    printf("\nArray after sorting:");
    for (int i = 0;i<z;i++)
    {
        printf("\t%d",arr[i]);
    }
    getch();
    return 0;
}

void insert(int arr[],int z)
{
    for(int i=1;i<z;i++)
    {

        int min = arr[i];
        int j = i-1;
        while(j>=0&&arr[j]>min)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1] = min;

    }
}
```

Output:



```
C:\Users\adnan>cd C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\Insertion

C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\Insertion>gcc insertion.c

C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\Insertion>a
Enter size of your array
6

Enter your array elements: 65 78 55 49 56 10

Array after sorting:    10      49      55      56      65      78
```