❖Insertion Sort in Descending order:

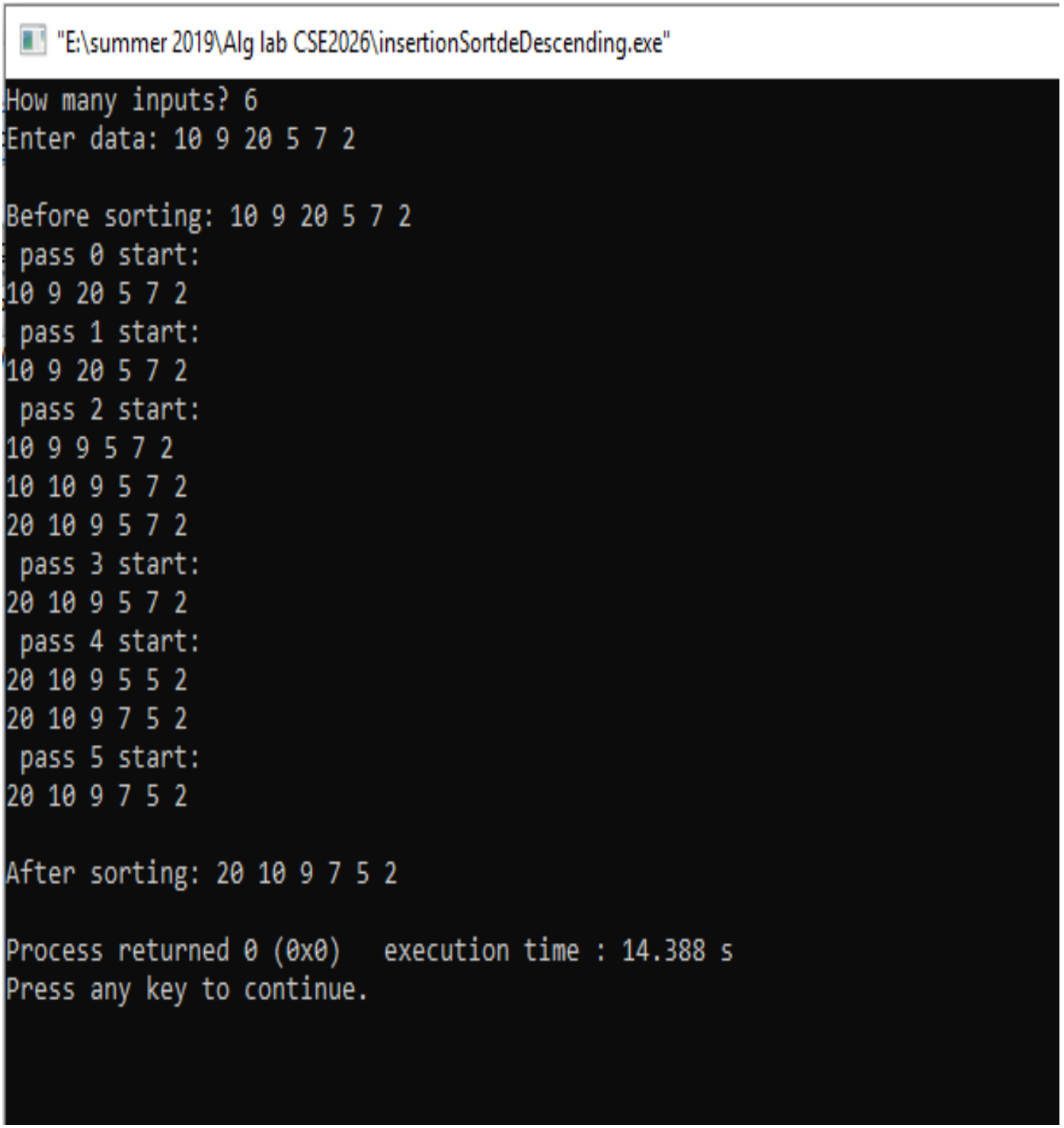
```
#include<stdio.h>
void input(int n, int arr[])
{
    printf("Enter data: ");
    int i;
    for(i=0; i<n; i++)
{
        scanf("%d", &arr[i]);
  }
}
void output(int n, int arr[])
{
    int i;
    for(i=0; i<n; i++)
  {
        printf("%d ", arr[i]);
    }
```

```
    printf("\n");
}

void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 0; i < n; i++) {
        key = arr[i];
        j = i - 1;
        printf(" pass %d start:\n",i);
      while (j >= 0 && arr[j] < key)
        {
            arr[j + 1] = arr[j];
            j --;
            output(n,arr);
        }
        arr[j + 1] = key;

        output(n,arr);
```

```
    }
}

int main()
{
    int arr[100], n;
    printf("How many inputs? ");
    scanf("%d", &n);
    input(n, arr);
    printf("\nBefore sorting: ");
    output(n, arr);
    insertionSort(arr, n);
    printf("\nAfter sorting: ");
    output(n, arr);
    return 0;
}
```

❖ OUTPUT:

Best case:8



Average case:11

```
How many inputs? 6
Enter data: 10 7 20 2 5 9

Before sorting: 10 7 20 2 5 9
 pass 0 start:
10 7 20 2 5 9
 pass 1 start:
10 7 20 2 5 9
 pass 2 start:
10 7 7 2 5 9
10 10 7 2 5 9
20 10 7 2 5 9
 pass 3 start:
20 10 7 2 5 9
 pass 4 start:
20 10 7 2 2 9
20 10 7 5 2 9
 pass 5 start:
20 10 7 5 2 2
20 10 7 5 5 2
20 10 7 7 5 2
20 10 9 7 5 2

After sorting: 20 10 9 7 5 2

Process returned 0 (0x0)   execution time : 15.503 s
Press any key to continue.
```

Worst case:15

```
Before sorting: 2 5 20 9 10 7
 pass 0 start:
2 5 20 9 10 7
 pass 1 start:
2 2 20 9 10 7
5 2 20 9 10 7
 pass 2 start:
5 2 2 9 10 7
5 5 2 9 10 7
20 5 2 9 10 7
 pass 3 start:
20 5 2 2 10 7
20 5 5 2 10 7
20 9 5 2 10 7
 pass 4 start:
20 9 5 2 2 7
20 9 5 5 2 7
20 9 9 5 2 7
20 10 9 5 2 7
 pass 5 start:
20 10 9 5 2 2
20 10 9 5 5 2
20 10 9 7 5 2

After sorting: 20 10 9 7 5 2

Process returned 0 (0x0)   execution time : 21.214 s
Press any key to continue.
```

❖ Result:

#Complexity of Insertion Sort:

Insertion sort runs in **O(n)** time in its best case and runs in **O(n^2)**in its worst and average cases.

Best case: O(n) It occurs when the data is in sorted order. After making one pass through the data and making no insertions, insertion sort exits.

Average case: O (n^2) since there is a wide variation with the running time.

Worst case: O(n^2) if the numbers were sorted in reverse order.

Insertion sort has a fast **best-case** running time and is a good sorting algorithm to use if the input list is already mostly sorted. For larger or more unordered lists, an algorithm with a faster **worst** and **average-case** running time.