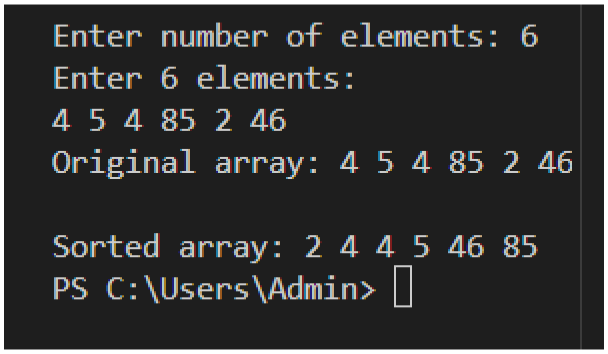
Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

```c
#include <stdio.h>



void swap(int *a, int *b) {

    int t = *a;

    *a = *b;

    *b = t;

}



int partition(int arr[], int low, int high) {

    int pivot = arr[high];

    int i = (low - 1);

    for (int j = low; j < high; j++) {

        if (arr[j] < pivot) {

            i++;

            swap(&arr[i], &arr[j]);

        }

    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```c
int main() {

    int n;

    printf("Enter number of elements: ");

    scanf("%d", &n);



    int arr[n];

    printf("Enter %d elements:\n", n);

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

        scanf("%d", &arr[i]);

    }



    printf("Original array: ");

    printArray(arr, n);



    quickSort(arr, 0, n - 1);



    printf("Sorted array: ");

    printArray(arr, n);

    return 0;
}
```

OUTPUT



```
Enter number of elements: 6
Enter 6 elements:
4 5 4 85 2 46
Original array: 4 5 4 85 2 46

Sorted array: 2 4 4 5 46 85
PS C:\Users\Admin> 
```