Description:

C program that implements Quick Sort, prints the array before/after sorting, and measures execution time using clock().

Source Code:

```
#include <stdio.h>

#include <time.h>

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

        int temp = *a;

        *a = *b;

        *b = temp;

}


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

        int pivot = arr[high]; // Pivot is the last element

        int i = (low - 1);          // Index of smaller element


        for (int j = low; j <= high - 1; j++) {

        // If current element is smaller than or equal to pivot

        if (arr[j] <= pivot) {

        i++; // Increment index of smaller element

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

        }

        }

        // Swap the pivot element with the element at the index i+1

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

```
        return (i + 1);
}
void quickSort(int arr[], int low, int high) {
        if (low < high) {
        // pIndex is partitioning index, arr[pIndex] is now at correct place
        int pIndex = partition(arr, low, high);

        // Separately sort elements before and after partition
        quickSort(arr, low, pIndex - 1);
        quickSort(arr, pIndex + 1, high);
        }
}
void printArray(int arr[], int size) {
        int i;
        for (i = 0; i < size; i++) {
        printf("%d ", arr[i]);
        }
        printf("\n");
}

int main() {
        int arr[] = {10, 7, 8, 9, 1, 5};

        clock_t start, end;
        double cpu_time_used;
```

```
    // Calculate the number of elements in the array
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Array:\n");
    printArray(arr, n);

    start = clock();

    // Call quickSort with the initial range (0 to n-1)
    quickSort(arr, 0, n - 1);

    end = clock();

    printf("\nSorted:\n");
    printArray(arr, n);

    // Calculate time
    cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;

    // Print execution time
    printf("Execution time: %.6f seconds\n", cpu_time_used);

    return 0;
}
```

Output:

```
Array:
10 7 8 9 1 5

Sorted:
1 5 7 8 9 10
Execution time: 0.000002 seconds
```