Experiment 18: Quick Sort

Aim:
To write a C program to arrange a series of numbers using Quick Sort.

Algorithm:

1. Start the program.
2. Choose a pivot element.
3. Partition the array around the pivot.
4. Recursively apply Quick Sort on left and right partitions.
5. Display the sorted array.
6. Stop.

Code:

```
#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);
    }
}
int main() {
    int arr[5] = {10, 7, 8, 9, 1}, n = 5;
    quickSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Sample Output:

```
Sorted array: 1 7 8 9 10

=== Code Execution Successful ===
```

Result:
The program successfully sorts numbers using Quick Sort.