# Step by Step Process

In Quick sort algorithm, partitioning of the list is performed using following steps...

- **Step 1 -** Consider the first element of the list as **pivot** (i.e., Element at first position in the list).
- **Step 2 -** Define two variables i and j. Set i and j to first and last elements of the list respectively.
- **Step 3 -** Increment i until list[i] > pivot then stop.
- **Step 4 -** Decrement j until list[j] < pivot then stop.
- **Step 5 -** If i < j then exchange list[i] and list[j].
- **Step 6 -** Repeat steps 3,4 & 5 until i > j.
- **Step 7 -** Exchange the pivot element with list[j] element.

Following is the sample code for Quick sort...

## Quick Sort Logic

```
//Quick Sort Logic
void quickSort(int list[10],int first,int last){
    int pivot,i,j,temp;

     if(first < last){
         pivot = first;
         i = first;
         j = last;

         while(i < j){
             while(list[i] <= list[pivot] && i < last)
                 i++;
             while(j>pivot && list[j] > list[pivot])
                 j--;
             if(i < j){
                  temp = list[i];
                  list[i] = list[j];
                  list[j] = temp;
             }
         }

         temp = list[pivot];
         list[pivot] = list[j];
```

```
            list[j] = temp;
            quickSort(list,first,j-1);
            quickSort(list,j+1,last);
        }
}
```

Consider the following unsorted list of elements...


Define pivot, left & right. Set pivot = 0, left = 1 & right = 7. Here '7' indicates 'size-1'.


Compare List[left] with List[pivot]. If **List[left]** is greater than **List[pivot]** then stop left otherwise move left to the next.
Compare List[right] with List[pivot]. If List[right] is smaller than List[pivot] then stop right otherwise move right to the previous.
Repeat the same until **left>=right**.
If both left & right are stoped but left<right then swap List[left] with List[right] and countinue the process.
If left>=right then swap List[pivot] with List[right].


Compare List[left]<List[pivot] as it is true increment left by one and repeat the same, left will stop at 8.
Compare List[right]>List[pivot] as it is true decrement right by one and repeat the same, right will stop at 2.


Here left & right both are stoped and left is not greater than right so we need to swap List[left] and List[right]


Compare List[left]<List[pivot] as it is true increment left by one and repeat the same, left will stop at 6.
Compare List[right]>List[pivot] as it is true decrement right by one and repeat the same, right will stop at 4.


Here left & right both are stoped and left is greater than right so we need to swap List[pivot] and List[right]


Here we can observe that all the numbers to the left side of 5 are smaller and right side are greater. That means 5 is placed in its correct position.
Repeat the same process on the left sublist and right sublist to the number 5.


In the left sublist as there are no smaller number than the pivot left will keep on moving to the next and stops at last number. As the List[right] is smaller, right stops at same position. Now left and right both are equal so we swap pivot with right.


In the right sublist left is grester than the pivot, left will stop at same position.
As the List[right] is greater than List[pivot], right moves towords left and stops at pivot number position.
Now left > right so we swap pivot with right. (6 is swap by itself).


Repeat the same recursively on both left and right sublists until all the numbers are sorted.
The final sorted list will be as follows...

# Complexity of the Quick Sort Algorithm

To sort an unsorted list with **'n'** number of elements, we need to make **((n-1)+(n-2)+(n-3)+.....+1) = (n (n-1))/2** number of comparisions in the worst case. If the list is already sorted, then it requires **'n'** number of comparisions.

**Worst Case : $O(n^2)$**
**Best Case : O (n log n)**
**Average Case : O (n log n)**