```c
#include <stdio.h>
#include <stdlib.h>

/*******************
This is code for quick sort
*******************/

// Get a random index in an array of length len
int randIdx(int len){
    return rand()%len;
}

// Swap two integers, of which, addresses are known
void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Print these many tabs for nice formatting
void printTabs(int numTabs){
    printf("\n");
    for(int i = 0; i < numTabs; i++)
        printf("\t");
}

// Print an array neatly indicating the pivot element in brackets
void print(int *arr, int len, int pivot){
    for(int i = 0; i < len; i++){
        if(i == pivot) printf("[");
        printf("%d", arr[i]);
        if(i == pivot) printf("]");
        if(i < len - 1) printf(" ");
    }
}

// Partition an array
// len: length of array
// pivot: index of pivot
// return the new index of the pivot element
int partition(int *arr, int len, int pivot){
    int pVal = arr[pivot]; // Value about which to be partitioned
    int *cpy = (int*)malloc(len * sizeof(int));
    int left = 0, right = len - 1;
    for(int i = 0; i < len; i++){
        if(arr[i] < pVal)
            cpy[left++] = arr[i];
        if(arr[i] > pVal)
            cpy[right--] = arr[i];
    }
    for(int i = left; i <= right; i++)
        cpy[i] = pVal;
    for(int i = 0; i < len; i++)
        arr[i] = cpy[i];
    free(cpy);
    return right;
}

// len: length of the array
// numTabs: nice formatting :)
void doQuickSort(int *arr, int len, int numTabs){
    if(len > 1){
        int pivot = randIdx(len); // Choose a random index as pivot
        printTabs(numTabs);
        printf("Before partition: ");
        print(arr, len, pivot);
        pivot = partition(arr, len, pivot);
        printTabs(numTabs);
        printf("After partition: ");
        print(arr, len, pivot);
        doQuickSort(arr, pivot, numTabs + 1);
        doQuickSort(arr + pivot + 1, len - pivot - 1, numTabs + 1);
        printTabs(numTabs);
        printf("After recursive sort: ");
        print(arr, len, pivot);
    }
}

int main(){
    int arr[10] = {8,9,6,3,4,2,1,5,10,7};
    printf("Initial: ");
    print(arr, 10, -1); // No pivot yet
    doQuickSort(arr, 10, 1);
    return 0;
}
```