Output :

Actual Array :

3 1 5 9 7 4 10 0

Sorted Array :

0 1 3 4 5 7 9 10.

Date 18/01/24 Expt. No. 02.

Expt. Name Page No. 4.

AIM: Implementation of Quick Sort using divide and Conquer Approach.

Algorithm:

I → Start
II → Take an array of size 'n' as input.
III → If value of n is 1 or less it is taken as sorted.
IV → Choose a pivoted element from the array (Set takes the last element).
V → Partition the array into two subarrays such that
- Elements less than pivot are in left subarray.
- Elements greater or equal to pivot are in right subarray.

VI → Recursively, apply quick Sort to the left and right subarray.
VII → Concatenate the arrays to get the final sorted array.
VIII → Stop.

Program Code:

```
#Include <stdio.h>
int MakePartition(int arr[], int left, int right){
    int pivot = arr[right];
    int i = left-1;
    for(int j= left ; j< right ; j++) {
        if (arr[j] < pivot) {
            i++;
```

Teacher's Signature:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 5

```
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
    int temp = arr[i+1];
    arr[i+1] = arr[right];
    arr[high] = temp;
    return i+1;
}
void quickSort(int arr[], int start, int end) {
    if (start < end) {
    int index = makePartition(arr, start, end);
      quickSort(arr, start, index-1);
      quickSort(arr, index+1, end);
    }
}
void print(int arr[], int n) {
    for (int i=0; i<n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int array[] = {3,1,5,9,7,4,10,0};
    int size = 8;
    printf("Actual Array: \n");
    printf(Array, 8);
    quickSort(array, 0, 7);
    printf("Sorted Array: \n");
    printArray(array, 8);
    return 0;
}
```

Teacher's Signature: _____