The code will compile using the **gcc version 4.6.1 (GCC)** or any other compiler that supports the standard.

Check the log files for result about the sorted data “testOuputLog.txt”

By default the program will read the data from the data.txt and sort them.
To test the program use QuickSort –test

Partition (A, L R): // A is the Array to be partitioned between the indices L and R (L<R)

```
Pivot = A[L] // Choose the leftmost element as the pivot
I = L + 1
For J from (L + 1) to R
    If (A[J] < Pivot)
        Swap(A[I] and A[J])
        I = I + 1
Swap(A[L] and A[I-1]) // Swap the pivot with element at index (I – 1)
Return (I – 1) // final index of the pivot
```

You need to understand this new Partition algorithm, **and submit your answers to the following questions:**

In terms of L and R, how many key comparisons, i.e. **(A[J] < Pivot),** are made by this Partition?

This is the total key comparison made when the algorithm uses LeftMost Pivot: **574782.** Hopefully you did not want us to show a recurrence relationship.

Draw a diagram, similar to the one I have for the text-book's Partition, BUT for this new partition algorithm: showing I and J and what the elements in the Array will "look like" (<= p, >= p, unexplored) after some iterations of the loop.

This example uses the same array in page 132.

Array: 5 3 1 9 8 2 4 7

5	**3ij**	1	9	8	2	4	7
5	3	**1ij**	9	8	2	4	7
5	3	1	**9ij**	8	2	4	7
5	3	1	**9i**	**8j**	2	4	7
5	3	1	**9i**	8	**2j**	4	7
5	3	1	2	**8i**	9	**4j**	7
5	3	1	2	4	9i	8	7j
4	**3ij**	1	2	5	9	8	7
4	3	**1ij**	2	5	9	8	7
4	3	1	**2ij**	5	9	8	7
2	**3ij**	1	4	5	9	8	7
2	**3i**	**1j**	4	5	9	8	7
1	2	3	4	5	**9**	**8ij**	7
1	2	3	4	5	**9**	8	**7ij**
1	2	3	4	5	**7**	**8ij**	9

After Quicksort: 1 2 3 4 5 7 8 9