Experiment 2

Problem Statement: Given a sequence of n numbers <a1, a2, ...,an>, find a permutation (reordering) <a1', a2', ...,an'> of the input sequence such that a1'<= a2'<=, ...,<=an', using quick sort algorithm.

Theory

Quicksort is an algorithm based on divide and conquer approach in which the array is split into subarrays and these sub-arrays are recursively called to sort the elements.

Algorithm

- ➔ Accept n, the number of elements
- ➔ Accept the data into array info
- ➔ l **<<<** 1, r **<<<** n
- ➔ i **<<<** l, j **<<<** r
- ➔ pivot **<<<** a[l]
- ➔ Loop while i<j
  - ◆ loop while a[i] < pivot
    - i **<<<** i+1
  - ◆ loop while a[j] > pivot
    - j **<<<** j -1
  - ◆ If i<j
    - Swap a(i) and a(j)
- ➔ If l<j
  - ◆ Repeat step 4 onwards taking r j -1
- ➔ If i<r
  - ◆ Repeat step 4 onwards taking l i + 1

Example

Consider an array of size 5

| 5 | 4 | 3 | 2 | 1 |

Pass 0: | 5 | 4 | 3 | 2 | 1 |
Pass 1: | 1 | 4 | 3 | 2 | 5 |
Pass 2: | 1 | 4 | 3 | 2 | 5 |
Pass 3: | 1 | 2 | 3 | 4 | 5 |

Sorted array: | 1 | 2 | 3 | 4 | 5 |

Complexity Analysis

Time Complexity

Assuming that n is 2m.
Best Case:
In the first pass we have n comparisons.
Then assuming we have 2 partitions with approx. n/2 elements in each, giving further n comparisons (approx.) This goes on till we are left with a partition of size 1. Since we have assumed n is 2m, thus the above procedure will go on m times & we would get a sum of approx m*n comparisons. But m = log2n, thus total number of comparisons is of the order of n(log n) i.e. **O(n log n)**.
Worst Case:
If the 1st pass results only in one partition i.e. say only left or only right (this will happen when the pivot is either the largest or smallest in the list), then we have n comparisons in the 1st pass & then n – 1 in the 2nd pass & n – 3 in the 3rd pass & so on…. Thus we have n – 1 passes with number of comparisons in each pass being n, n -1, n – 2, n – 3, ……. Thus the sum of comparisons is an arithmetic progression and hence is order of n2
i.e. O(n2).

Space Complexity
The best- and average-case space-complexity is O(log(N)), for the stack-space used. The worst-case space-complexity is O(N), but it can be limited to O(log(N)).

Code

```
#include <stdio.h>
#define MAX 100
int data[MAX], n, i, pass = 0;

void printArray(int array[])
{
   for (i = 0; i < n; i++)
   {
      printf("| %d |", array[i]);
   }
   printf("\n");
}

void swap(int array[], int c, int d)
{
   int temp = array[c];
   array[c] = array[d];
   array[d] = temp;
}

int partition(int array[], int l, int m)
{
   int k, j;
   int pivot = array[m];
   k = l - 1;
   for (j = l; j < m; j++)
   {
      if (array[j] < pivot)
      {
         k++;
         swap(array, k, j);
      }
   }
   swap(array, k + 1, m);
   return k + 1;
}
```

```
void quickSort(int array[], int l, int r)
{
   int pi;

   if (l < r)
   {
      printf("Pass %d: ", pass++);
      printArray(array);
      printf("\n");
      int pi = partition(array, l, r);
      quickSort(array, l, pi - 1);
      quickSort(array, pi + 1, r);
   }
}

int main()
{
   printf("\n");
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Start entering %d integers: \n", n);
   for (i = 0; i < n; i++)
   {
      printf("data[%d] = ", i);
      scanf("%d", &data[i]);
   }

   printf("\n------- Array before sorting -------\n");
   printArray(data);
   printf("------- x ------- x ------- x -------\n\n");

   quickSort(data, 0, n - 1);

   printf("\n------- Array after sorting -------\n");
   printArray(data);
   printf("------- x ------- x ------- x -------\n\n");

   return 0;
}
```

Output

```
Enter number of elements: 7
Start entering 7 integers:
data[0] = 7
data[1] = 3
data[2] = 2
data[3] = 5
data[4] = 4
data[5] = 1
data[6] = 6
```

```
------- Array before sorting -------
| 7 || 3 || 2 || 5 || 4 || 1 || 6 |
------- x ------- x ------- x -------

Pass 0: | 7 || 3 || 2 || 5 || 4 || 1 || 6 |

Pass 1: | 3 || 2 || 5 || 4 || 1 || 6 || 7 |

Pass 2: | 1 || 2 || 5 || 4 || 3 || 6 || 7 |

Pass 3: | 1 || 2 || 3 || 4 || 5 || 6 || 7 |


------- Array after sorting -------
| 1 || 2 || 3 || 4 || 5 || 6 || 7 |
------- x ------- x ------- x -------
```