Randomized Quick Sort

Below is the C++ implementation used (`1.cpp`), followed by the program output and an embedded screenshot of the run.

Code

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int comparisons = 0; // global counter for comparisons

// Function to swap two elements
void swap(int &a, int &b) {
int temp = a;
a = b;
b = temp;
}

// Partition function with randomized pivot
int partition(int arr[], int low, int high) {
// Randomly choose a pivot index between low and high
int pivotIndex = low + rand() % (high - low + 1);
```

```
swap(arr[pivotIndex], arr[high]); // move pivot to end


int pivot = arr[high];
int i = low - 1;


for (int j = low; j < high; j++) {
comparisons++; // each check arr[j] <= pivot is a comparison
if (arr[j] <= pivot) {
i++;
swap(arr[i], arr[j]);
}
}
swap(arr[i + 1], arr[high]);
return i + 1;
}


// QuickSort function
void randomizedQuickSort(int arr[], int low, int high) {
if (low < high) {
int pi = partition(arr, low, high);


randomizedQuickSort(arr, low, pi - 1);
randomizedQuickSort(arr, pi + 1, high);
```

```
}

}


int main() {

srand(time(0)); // seed for random pivot


int n;

cout << "Enter number of elements: ";

cin >> n;


int arr[n];

cout << "Enter " << n << " elements:\n";

for (int i = 0; i < n; i++)

cin >> arr[i];


randomizedQuickSort(arr, 0, n - 1);


cout << "\nSorted array: ";

for (int i = 0; i < n; i++)

cout << arr[i] << " ";


cout << "\nTotal comparisons: " << comparisons << endl;
```

```
return 0;

}
```

Output