```
template <class elemType>
void quickSort(elemType list[], int length, int& noc, int& nom)
{
  noc = 0;
  nom = 0;
  recQuickSort(list, 0, length-1, noc, nom);
}

template <class elemType>
void recQuickSort(elemType list[], int first, int last, int& noc, int& nom)
{
  int pLoc;
  if (first < last)
  {
    pLoc = partition(list, first, last, noc, nom);
    recQuickSort(list, first, pLoc-1, noc, nom);
    recQuickSort(list, first, pLoc+1, noc, nom);
  }
}

#include "searchSortAlgorithms.h"
#include <iostream>

int main()
{
  cout << "\n Quick Sort" << endl;
  list<int> myList{32,11,65,11,50,37,24,63,50};
  int noc=0;
  int nom=0;
  quickSort(myList, 9, noc, nom);
  cout << "The number of comparisons: " << noc << endl;
  cout << "The number of data movements: " << nom << endl;
}
```

Emre Koyo
2000006713

```
template <class elemType>
int partition (elemType list[], int first, int last, int& noc, int& nom)
{
    elemType p;
    int ind;
    int sind;
    swap (list, first, (first+last)/2);
    nom = nom + 3;
    p = list[first];
    sind = first;
    for (ind = first+1; ind <= last; ind++)
    {   if (list[ind] < p)
        {
            sind = sind + 1;
            swap (list, sind, ind);
            nom = nom + 3;
        }
    }
    swap (list, first, sind);
    nom = nom + 3;
    return sind;
}

template <class elemType>
void swap (elemType list[], int first, int second)
{
    elemType temp;
    temp = list[first];
    list[first] = list[second];
    list[second] = temp;
}
```

Emre Kaya
2000006713