**Divide and Conquer Sorting Algorithms**

**Merge Sort Algorithm**

void mergeSort( ListType list[], int first, int last )

if ( first != last )

int middle = ( first + last ) / 2;

mergeSort( list, first, middle );

mergeSort( list, middle + 1, last );

merge( list, first, middle, last );

void merge( ListType list[], int first, int middle, int last )

ListType temp[ LIST\_MAX ];

int f1 = first, l1 = middle;

int f2 = middle + 1, l2 = last;

int fTemp = f1;

// comparing front values of two smaller lists – moving smaller value to temp list

while ( ( f1 <= l1 ) && ( f2 <= l2 ) )

if ( list[ f1 ] < list[ f2 ] )

temp[ fTemp ] = list[ f1 ];

++f1;

else

temp[ fTemp ] = list[ f2 ];

++f2

++fTemp;

// moves remaining left list values to temp list

for ( int i = f1; i <= l1; ++i )

temp[ fTemp ] = list[ i ];

++fTemp;

// moves remaining right list value to temp list

for ( int i = f2; i <= l2; ++i )

temp[ fTemp ] = list[ i ];

++fTemp;

// moves sorted temp list back to original list

for ( int i = first; i <= last; ++i )

list[ i ] = temp[ i ];

**Quick Sort Algorithm**

void quickSort( ListType list[], int left, int right )

if ( left < right )

int pivotPos = partition( list, left, right );

quickSort ( list, left, pivotPos – 1 );

quickSort ( list, pivotPos + 1, right );

int partition( ListType list[], int left, int right ) **// First Element Pivot Scheme**

ListType pivot;

int lastMin;

pivot = list[ left ];

lastMin = left;

for ( int i = left + 1; i <= right; ++i )

if ( list[ i ] < pivot )

++lastMin;

swap( list[ lastMin ], list[ i ] );

swap( list[ left ], list[ lastMin ] )

return lastMin;

int partition( ListType list[], int left, int right ) **// Middle Element Pivot Scheme**

ListType pivot;

int lastMin, middle;

**middle = ( left + right ) / 2;**

**swap( list[ left ], list[ middle ] );**

pivot = list[ left ];

lastMin = left;

for ( int i = left + 1; i <= right; ++i )

if ( list[ i ] < pivot )

++lastMin;

swap( list[ lastMin ], list[ i ] );

swap( list[ left ], list[ lastMin ] )

return lastMin;