public void MergeSort( int[] arrayToSort)

{

    if( arrayToSort != null && arrayToSort.Length > 1)

    {

        int[] tempArray = new int[arrayToSort.Length];

        MergeSort\_Internal(arrayToSort, 0, arrayToSort.Length - 1, tempArray);

    }

}

private void MergeSort\_Internal( int[] arrayToSort, int start, int end,

                                 int[] tempArray )

{

    if( first < end)

    {

        int middle = (start + end) / 2;

       MergeSort\_Internal(arrayToSort, start, middle, tempArray);

       MergeSort\_Internal(arrayToSort, middle+1, end, tempArray);

MergeLists(arrayToSort, start, middle, middle+1, end, tempArray);

    }

}

 private void MergeLists(    int [] arrayToSort,

                            int firstLeft, int lastLeft,

                            int firstRight, int lastRight,

                            int[] tempArray )

{

    // we'll merge things into the temp array to sort them

    // then copy them back into arrayToSort

    int locationInTemp = 0;

    int currentFirstLeft = firstLeft;

    int currentFirstRight = firstRight;

    bool itemsRemainingOnLeft = currentFirstLeft <= lastLeft;

    bool itemsRemainingOnRight = currentFirstRight <= lastRight;

    // while we have anything remaining in either sublist

    while(itemsRemainingOnLeft  || itemsRemainingOnRight)

    {

        // if we've got two items - one from each sublist

        if( itemsRemainingOnLeft  && itemsRemainingOnRight)

        {

            // the smaller item goes into the merge-list next

if(arrayToSort[currentFirstLeft]<arrayToSort[currentFirstRight])

            {

tempArray[locationInTemp] = arrayToSort[currentFirstLeft];

                locationInTemp++;

                currentFirstLeft++;

            }  
 // ‘else’ is on the next page

            else

            {

                // this does the same thing as the 'true' case,

                //  except: this is less code, and

// this is for the right side

tempArray[locationInTemp++] = arrayToSort[currentFirstRight++];

            }

        }

        else if( itemsRemainingOnLeft)

        {

            // if we've run out of items on the right, then

            // copy all remaining items from the left into the

// merge list

tempArray[locationInTemp++] = arrayToSort[currentFirstLeft++];

        }

        else // itemsRemainingOnRight == true

        {

tempArray[locationInTemp++] = arrayToSort[currentFirstRight++];

        }

        itemsRemainingOnLeft = currentFirstLeft <= lastLeft;

        itemsRemainingOnRight = currentFirstRight <= lastRight;

    } // End of 'merging two subarrays into tempArray' loop

    // Now copy the data from the temp array back to the

    // original slice of the array we're sorting

    int howManyToCopy = lastRight - firstLeft + 1;

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

    {

        arrayToSort[firstLeft + i] = tempArray[i];

    }

}