Description of the algorithm

Conceptually, a merge sort works as follows

Merge sort incorporates two main ideas to improve its runtime:

Example: Using merge sort to sort a list of integers contained in an array:

Suppose we have an array A with n indices ranging from A0 to An − 1. We apply merge sort to A(A0..Ac − 1) and A(Ac..An − 1) where c is the integer part of n / 2. When the two halves are returned they will have been sorted. They can now be merged together to form a sorted array.

In a simple pseudocode form, the algorithm could look something like this:

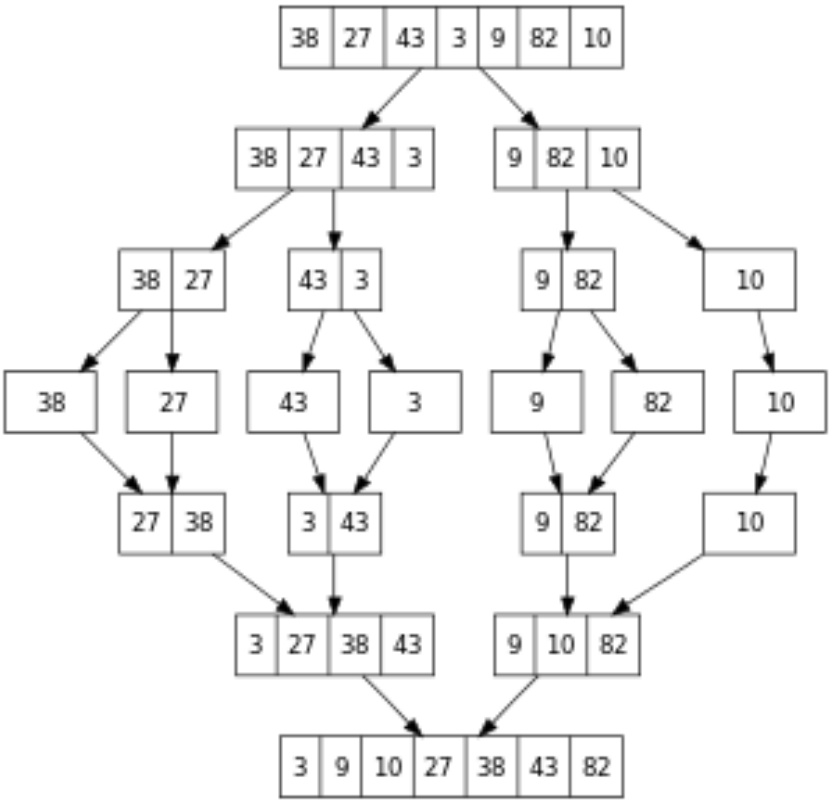
```
function merge_sort(m)
   if length(m) ≤ 1
      return m
   var list left, right, result
   var integer middle = length(m) / 2
   for each x in m up to middle
       add x to left
   for each x in m after middle
       add x to right
   left = merge_sort(left)
   right = merge_sort(right)
   result = merge(left, right)
   return result
```

Following writing merge_sort function, then it is required to merge both the left and right lists created above. There are several variants for the merge() function; one possibility is this:

```
function merge(left,right)
   var list result
   while length(left) > 0 or length(right) > 0
      if length(left) > 0 and length(right) > 0
         if first(left) ≤ first(right)
```

```
            append first(left) to result
            left = rest(left)
        else
            append first(right) to result
            right = rest(right)
    else if length(left) > 0
        append first(left) to result
        left = rest(left)
    else if length(right) > 0
        append first(right) to result
        right = rest(right)
end while
return result
```

Marking Scheme



C# code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Merge_Sort
{
    class Program
    {
        /// <summary>
        /// Merge Sort function
        /// </summary>
        /// <param name="input"></param>
        /// <param name="left"></param>
        /// <param name="right"></param>
        public static void Merge_Sort(int[] input, int left, int right)
        {
            if (left < right)
            {
                //calculate middle index of array
                int middle = (left + right) / 2;
                //call function Merge_Sort
                Merge_Sort(input, left, middle);
                Merge_Sort(input, middle + 1, right);

                //Merge
                int[] leftArray = new int[middle - left + 1];
                int[] rightArray = new int[right - middle];
                //copyt array
                Array.Copy(input, left, leftArray, 0, middle - left + 1);
                Array.Copy(input, middle + 1, rightArray, 0, right - middle);

                int indexI = 0;
                int indexJ = 0;

                //loop array
                for (int k = left; k < right + 1; k++)
                {
                    if (indexI == leftArray.Length)
                    {
                        //add value to right
                        input[k] = rightArray[indexJ];
                        indexJ++;
                    }
                    else if (indexJ == rightArray.Length)
                    {
                        //add value to left
                        input[k] = leftArray[indexI];
                        indexI++;
                    }
                    else if (leftArray[indexI] <= rightArray[indexJ])
                    {
                        //add value to left
                        input[k] = leftArray[indexI];
                        indexI++;
                    }
                    else
                    {
```

```
                    //add value to right
                    input[k] = rightArray[indexJ];
                    indexJ++;
                }
            }
        }
    }
    /// <summary>
    /// Main function
    /// </summary>
    /// <param name="args"></param>
    static void Main(string[] args){
        Console.Write("Enter number of elements: ");
        int max = Convert.ToInt32(Console.ReadLine());
        //array for inputed values
        int[] numbers = new int[max];
        ////promt user to enter values
        for (int i = 0; i < max; i++){
             Console.Write("Enter [" + (i + 1).ToString() + "] element: ");
             numbers[i] = Convert.ToInt32(Console.ReadLine());
        }
        //show all values
        Console.Write("Array:");
        Console.Write("\n");
        for (int k = 0; k < max; k++)
        {
            //print values
            Console.Write(numbers[k] + " ");
        }

        Console.WriteLine("\n\nArray after Merge Sort");
        //call function Merge Sort
        Merge_Sort(numbers, 0, max - 1);
        for (int i = 0; i < max; i++)
        {
            Console.Write(numbers[i] + " ");
        }
        //delay
        Console.ReadLine();

    }

}
}
```