class SingleLinkedList

{

private Node start;

public void DisplayList()

{

Node p;

if (start == null)

{

Console.WriteLine("List is Empty");

return;

}

Console.Write("List is : ");

p = start;

while (p != null)

{

Console.Write(p.info + " ");

p = p.link;

}

Console.WriteLine();

} //

public void InsertAtEnd(int data)

{

Node p;

Node temp = new Node(data);

if (start == null)

{

start = temp;

return;

}

p = start;

while (p.link != null)

p = p.link;

p.link = temp;

} //

public void CreateList()

{

int i, n, data;

Console.Write("Enter the number of nodes : ");

n = Convert.ToInt32(Console.ReadLine());

if (n == 0)

return;

for (i = 1; i <= n; i++)

{

Console.Write("Enter the element to be inserted : ");

data = Convert.ToInt32(Console.ReadLine());

InsertAtEnd(data);

}

} //

public void BubbleSortExData()

{

Node end, p, q;

for (end = null; end != start.link; end = p)

{

for (p = start; p.link != end; p = p.link)

{

q = p.link;

if (p.info > q.info)

{

int temp = p.info;

p.info = q.info;

q.info = temp;

}

}

}

} //

public SingleLinkedList Merge1(SingleLinkedList list)

{

SingleLinkedList mergeList = new SingleLinkedList();

mergeList.start = Merge1(start, list.start);

return mergeList;

} //

private Node Merge1(Node p1, Node p2)

{

Node startM;

if (p1.info <= p2.info)

{

startM = new Node(p1.info);

p1 = p1.link;

}

else

{

startM = new Node(p2.info);

p2 = p2.link;

}

Node pM = startM;

while (p1 != null && p2 != null)

{

if (p1.info <= p2.info)

{

pM.link = new Node(p1.info);

p1 = p1.link;

}

else

{

pM.link = new Node(p2.info);

p2 = p2.link;

}

pM = pM.link;

}

//if second list has finished and elements left in first list

while (p1 != null)

{

pM.link = new Node(p1.info);

p1 = p1.link;

pM = pM.link;

}

//if first list has finished and elements left in second list

while (p2 != null)

{

pM.link = new Node(p2.info);

p2 = p2.link;

pM = pM.link;

}

return startM;

} //

public SingleLinkedList Merge2(SingleLinkedList list)

{

SingleLinkedList mergeList = new SingleLinkedList();

mergeList.start = Merge2(start, list.start);

return mergeList;

} //

private Node Merge2(Node p1, Node p2)

{

Node startM;

if (p1.info <= p2.info)

{

startM = p1;

p1 = p1.link;

}

else

{

startM = p2;

p2 = p2.link;

}

Node pM = startM;

while (p1 != null && p2 != null)

{

if (p1.info <= p2.info)

{

pM.link = p1;

pM = pM.link;

p1 = p1.link;

}

else

{

pM.link = p2;

pM = pM.link;

p2 = p2.link;

}

}

if (p1 == null)

pM.link = p2;

else

pM.link = p1;

return startM;

} //

}

}

class Node

{

public int info;

public Node link;

public Node(int i)

{

info = i;

link = null;

}

}

}

class Program

{

static void Main(string[] args)

{

SingleLinkedList list1 = new SingleLinkedList();

SingleLinkedList list2 = new SingleLinkedList();

list1.CreateList();

list2.CreateList();

list1.BubbleSortExData();

list2.BubbleSortExData();

Console.WriteLine("First list - "); list1.DisplayList();

Console.WriteLine("Second list - "); list2.DisplayList();

SingleLinkedList list3;

list3 = list1.Merge1(list2); // merging two sorted lists

Console.WriteLine("Merged list - "); list3.DisplayList();

Console.WriteLine("First list - "); list1.DisplayList(); //original lists will not be changed by Merge1 method

Console.WriteLine("Second list - "); list2.DisplayList();

list3 = list1.Merge2(list2); //////merge by re arranging links

Console.WriteLine("Merged list - "); list3.DisplayList();

Console.WriteLine("First list - "); list1.DisplayList(); //original lists will be changed by Merge2 method

Console.WriteLine("Second list - "); list2.DisplayList();

}

}

}