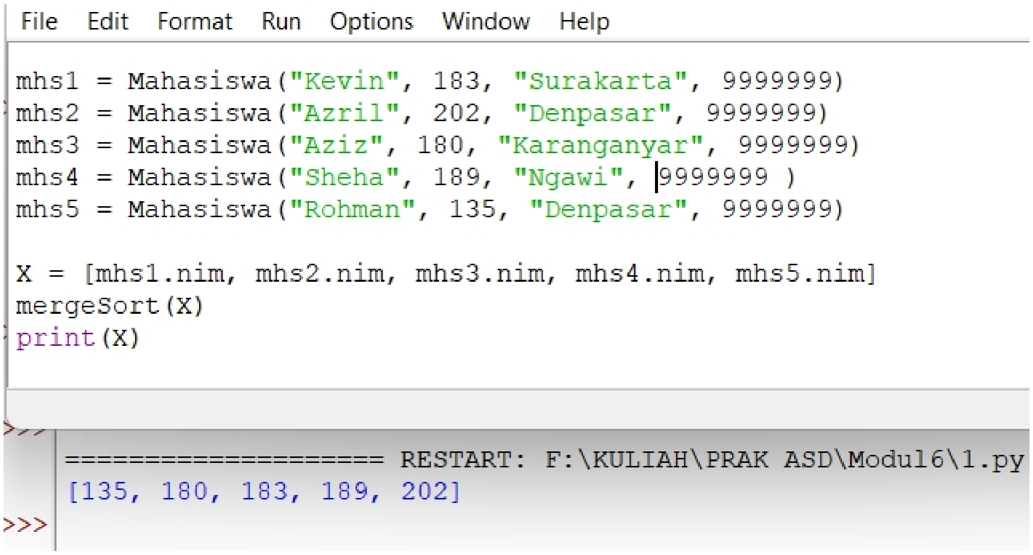
NAMA : KEVIN AVICENNA WIDIARTO
NIM : L200200183
Modul : 6

Praktikum Algoritma Struktur Data

MODUL 6

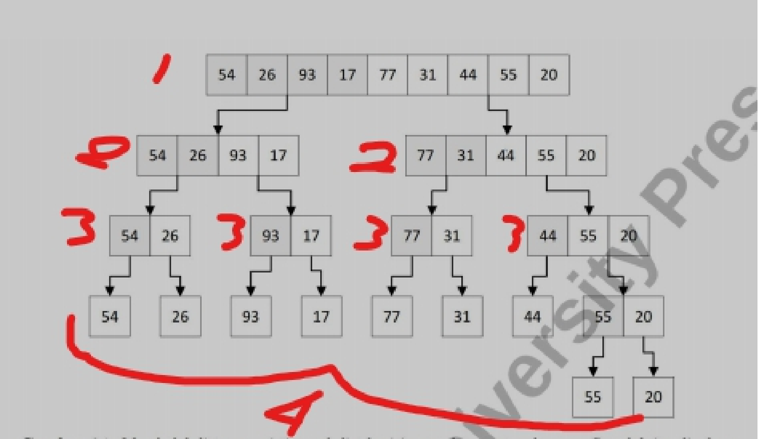
No 1



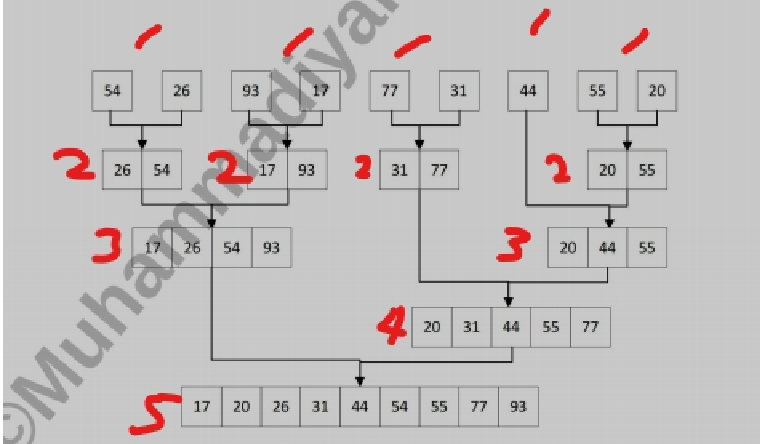
```
mhs1 = Mahasiswa("Kevin", 183, "Surakarta", 9999999)
mhs2 = Mahasiswa("Azril", 202, "Denpasar", 9999999)
mhs3 = Mahasiswa("Aziz", 180, "Karanganyar", 9999999)
mhs4 = Mahasiswa("Sheha", 189, "Ngawi", 9999999 )
mhs5 = Mahasiswa("Rohman", 135, "Denpasar", 9999999)

X = [mhs1.nim, mhs2.nim, mhs3.nim, mhs4.nim, mhs5.nim]
mergeSort(X)
print(X)
```

```
>>>
==================== RESTART: F:\KULIAH\PRAK ASD\Modul6\1.py
[135, 180, 183, 189, 202]
>>>
```

No 2

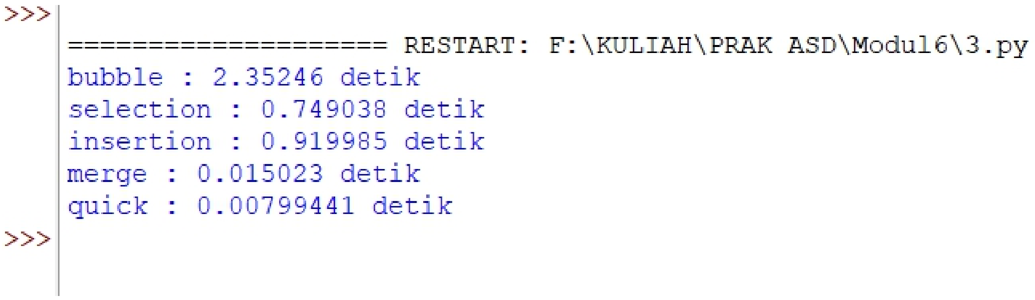


Gambar 6.1: Membelah list sampai tiap sub-list berisi satu elemen atau kosong. Sesudah itu digabung seperti ditunjukkan di Gambar 6.2.



Gambar 6.2: Menggabungkan list satu demi satu.

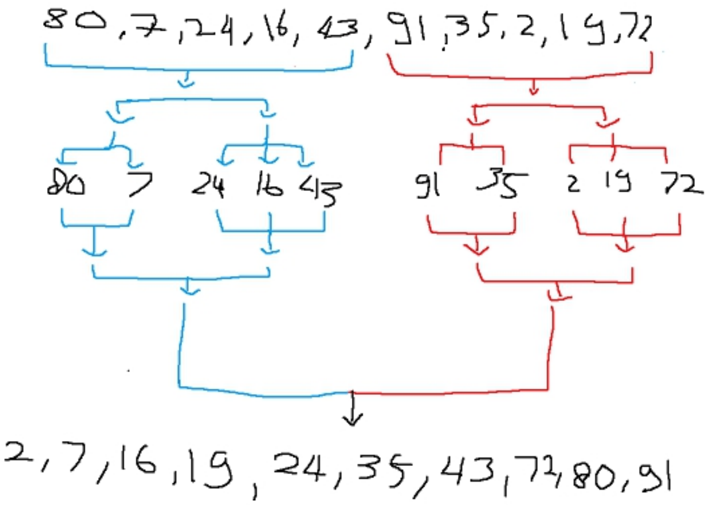
No 3



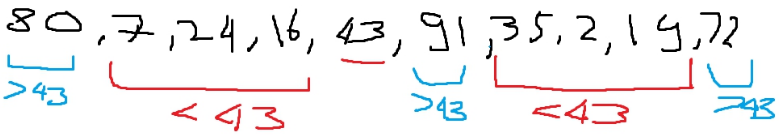
```
>>>
    ==================== RESTART: F:\KULIAH\PRAK ASD\Modul6\3.py
    bubble : 2.35246 detik
    selection : 0.749038 detik
    insertion : 0.919985 detik
    merge : 0.015023 detik
    quick : 0.00799441 detik
>>>
```

No 4
L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72],

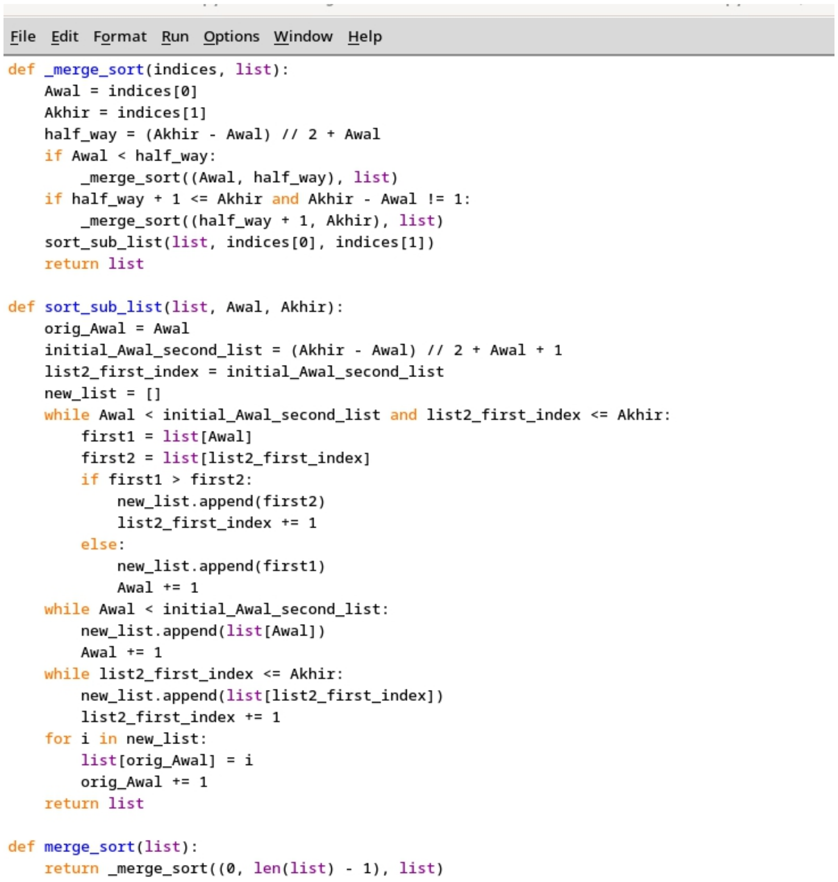
MergeSort



QuickSort

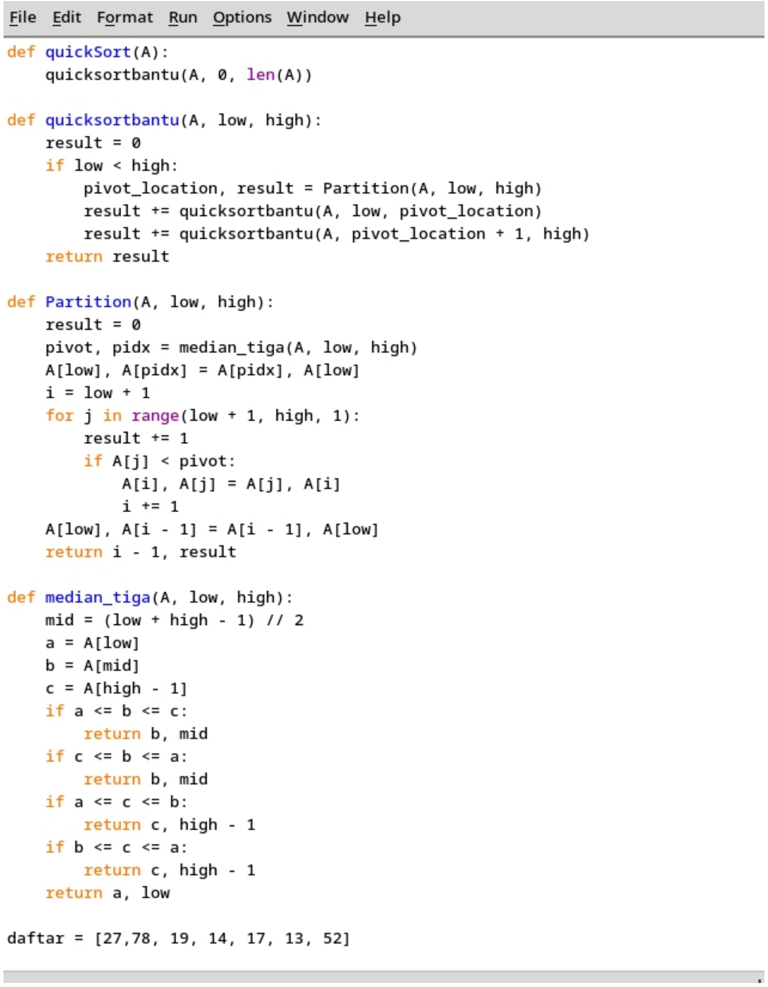


No 5



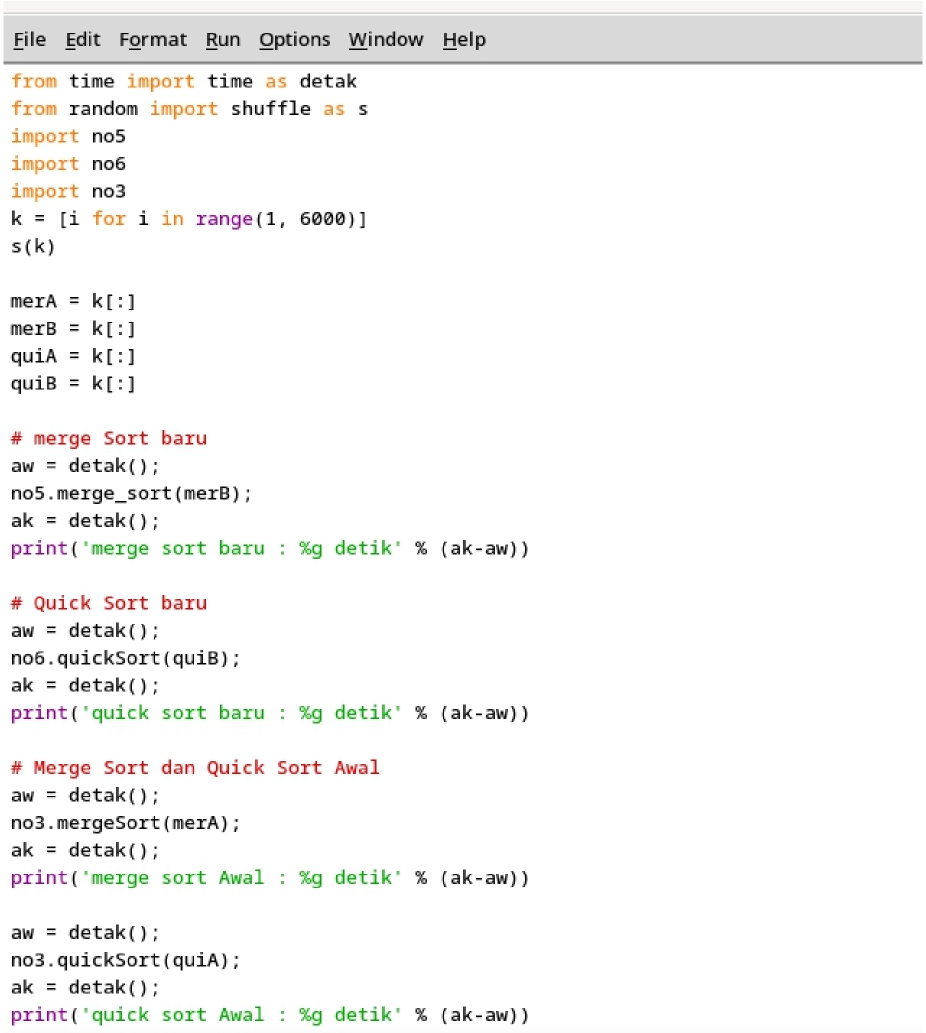
```
def _merge_sort(indices, list):
    Awal = indices[0]
    Akhir = indices[1]
    half_way = (Akhir - Awal) // 2 + Awal
    if Awal < half_way:
        _merge_sort((Awal, half_way), list)
    if half_way + 1 <= Akhir and Akhir - Awal != 1:
        _merge_sort((half_way + 1, Akhir), list)
    sort_sub_list(list, indices[0], indices[1])
    return list

def sort_sub_list(list, Awal, Akhir):
    orig_Awal = Awal
    initial_Awal_second_list = (Akhir - Awal) // 2 + Awal + 1
    list2_first_index = initial_Awal_second_list
    new_list = []
    while Awal < initial_Awal_second_list and list2_first_index <= Akhir:
        first1 = list[Awal]
        first2 = list[list2_first_index]
        if first1 > first2:
            new_list.append(first2)
            list2_first_index += 1
        else:
            new_list.append(first1)
            Awal += 1
    while Awal < initial_Awal_second_list:
        new_list.append(list[Awal])
        Awal += 1
    while list2_first_index <= Akhir:
        new_list.append(list[list2_first_index])
        list2_first_index += 1
    for i in new_list:
        list[orig_Awal] = i
        orig_Awal += 1
    return list

def merge_sort(list):
    return _merge_sort((0, len(list) - 1), list)
```

No 6

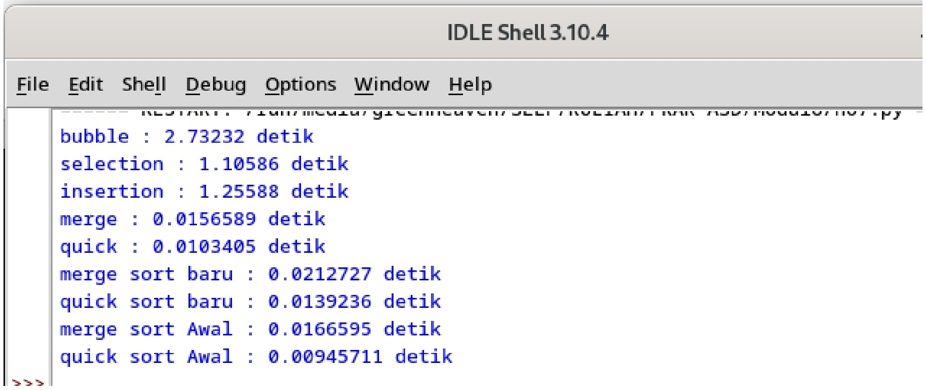


```
def quickSort(A):
    quicksortbantu(A, 0, len(A))

def quicksortbantu(A, low, high):
    result = 0
    if low < high:
        pivot_location, result = Partition(A, low, high)
        result += quicksortbantu(A, low, pivot_location)
        result += quicksortbantu(A, pivot_location + 1, high)
    return result

def Partition(A, low, high):
    result = 0
    pivot, pidx = median_tiga(A, low, high)
    A[low], A[pidx] = A[pidx], A[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if A[j] < pivot:
            A[i], A[j] = A[j], A[i]
            i += 1
    A[low], A[i - 1] = A[i - 1], A[low]
    return i - 1, result

def median_tiga(A, low, high):
    mid = (low + high - 1) // 2
    a = A[low]
    b = A[mid]
    c = A[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low

daftar = [27,78, 19, 14, 17, 13, 52]
```

No 7

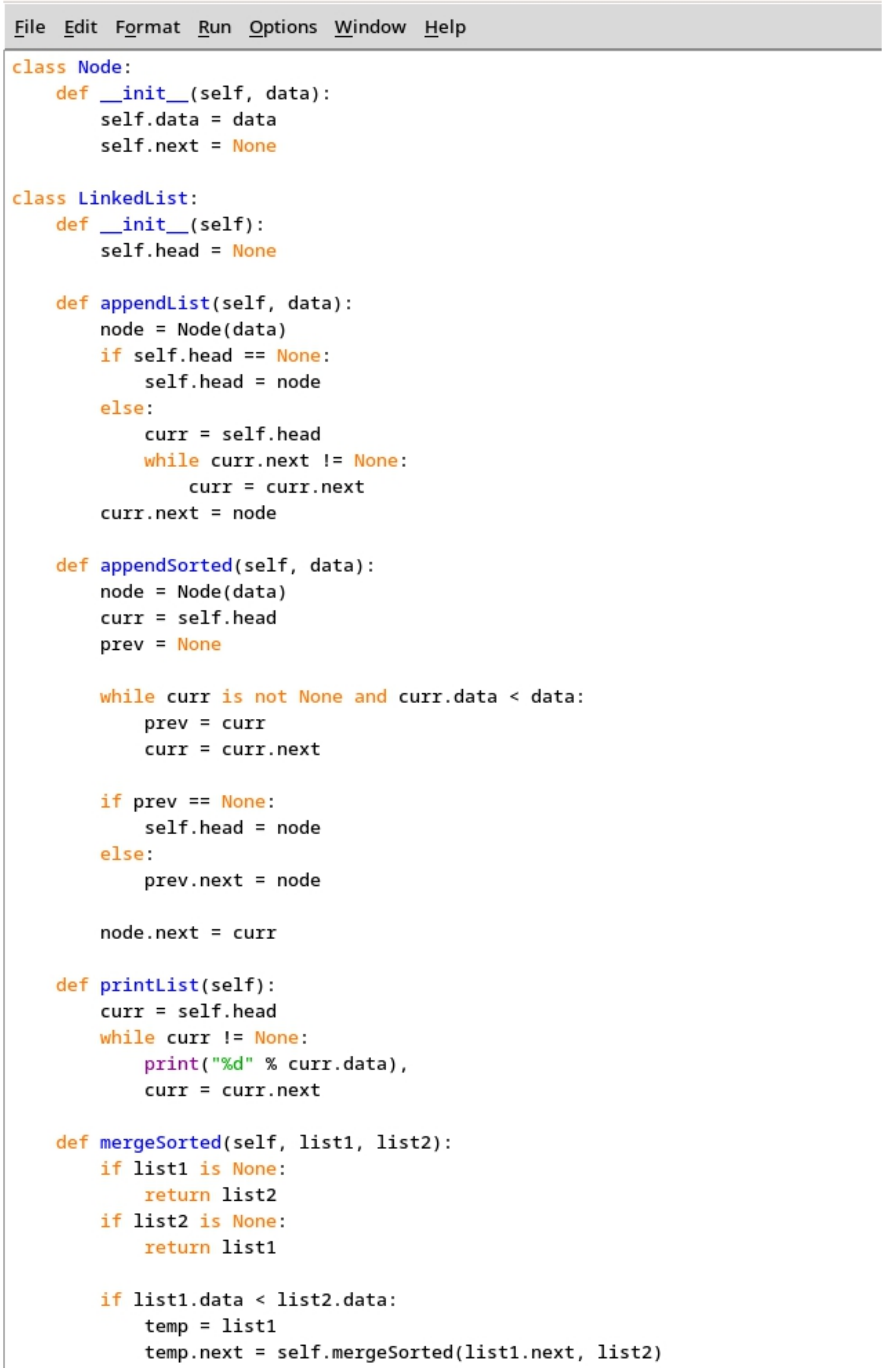


```
from time import time as detak
from random import shuffle as s
import no5
import no6
import no3
k = [i for i in range(1, 6000)]
s(k)

merA = k[:]
merB = k[:]
quiA = k[:]
quiB = k[:]

# merge Sort baru
aw = detak();
no5.merge_sort(merB);
ak = detak();
print('merge sort baru : %g detik' % (ak-aw))

# Quick Sort baru
aw = detak();
no6.quickSort(quiB);
ak = detak();
print('quick sort baru : %g detik' % (ak-aw))

# Merge Sort dan Quick Sort Awal
aw = detak();
no3.mergeSort(merA);
ak = detak();
print('merge sort Awal : %g detik' % (ak-aw))

aw = detak();
no3.quickSort(quiA);
ak = detak();
print('quick sort Awal : %g detik' % (ak-aw))
```

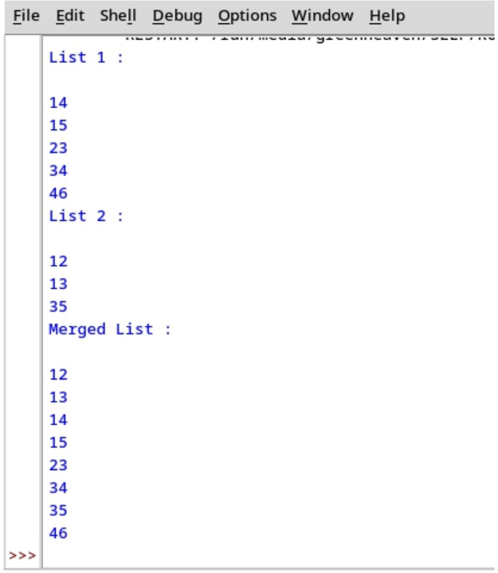


```
====== RESTART: /run/media/greenheaven/SELF/KULIAH/PRAK ASD/Modul6/no7.py =
bubble : 2.73232 detik
selection : 1.10586 detik
insertion : 1.25588 detik
merge : 0.0156589 detik
quick : 0.0103405 detik
merge sort baru : 0.0212727 detik
quick sort baru : 0.0139236 detik
merge sort Awal : 0.0166595 detik
quick sort Awal : 0.00945711 detik
>>>
```

No 8



```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def appendList(self, data):
        node = Node(data)
        if self.head == None:
            self.head = node
        else:
            curr = self.head
            while curr.next != None:
                curr = curr.next
        curr.next = node

    def appendSorted(self, data):
        node = Node(data)
        curr = self.head
        prev = None

        while curr is not None and curr.data < data:
            prev = curr
            curr = curr.next

        if prev == None:
            self.head = node
        else:
            prev.next = node

        node.next = curr

    def printList(self):
        curr = self.head
        while curr != None:
            print("%d" % curr.data),
            curr = curr.next

    def mergeSorted(self, list1, list2):
        if list1 is None:
            return list2
        if list2 is None:
            return list1

        if list1.data < list2.data:
            temp = list1
            temp.next = self.mergeSorted(list1.next, list2)
    def mergeSorted(self, list1, list2):
        if list1 is None:
            return list2
        if list2 is None:
            return list1

        if list1.data < list2.data:
            temp = list1
            temp.next = self.mergeSorted(list1.next, list2)
        else:
            temp = list2
            temp.next = self.mergeSorted(list1, list2.next)
        return temp


list1 = LinkedList()
list1.appendSorted(15)
list1.appendSorted(14)
list1.appendSorted(34)
list1.appendSorted(23)
list1.appendSorted(46)

print("List 1 :"),
list1.printList()

list2 = LinkedList()
list2.appendSorted(35)
list2.appendSorted(13)
list2.appendSorted(12)

print("List 2 :"),
list2.printList()

list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)

print("Merged List :"),
list3.printList()
```



```
List 1 :

14
15
23
34
46
List 2 :

12
13
35
Merged List :

12
13
14
15
23
34
35
46
>>>
```