Nama    : Rahmat Beny Susanto

NIM      : L200180079

Kelas    : C

PRAKTIKUM ASD

MODUL 6

TUGAS

1. Mengubah kode mergeSort dan quicksort agar bisa mengurutkan list yang berisi objectobject MhsTIF. Dari class MhsTIF

```
class MhsTIF():
    def __init__(self, nim):
        self.nim = nim

    def __str__(self):
        return str(self.nim)

c0 = MhsTIF(10)
c1 = MhsTIF(51)
c2 = MhsTIF(2)
c3 = MhsTIF(18)
c4 = MhsTIF(4)
c5 = MhsTIF(31)
c6 = MhsTIF(13)
c7 = MhsTIF(5)
c8 = MhsTIF(23)
c9 = MhsTIF(64)
c10 = MhsTIF(29)

c0.next = c1
c1.next = c2
c2.next = c3
c3.next = c4
c4.next = c5
c5.next = c6
c6.next = c7
c7.next = c8
c8.next = c9
c9.next = c10

##MergeSort
def mergeSort(A):
    #print("Membelah       ",A)
    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]

        mergeSort(separuhkiri)
        mergeSort(separuhkanan)

        i = 0;j=0;k=0
        while i < len(separuhkiri) and j < len(separuhkanan):
            if separuhkiri[i] < separuhkanan[j]:
                A[k] = separuhkiri[i]
                i = i + 1
            else:
                A[k] = separuhkanan[j]
                j = j + 1
            k=k+1

        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i = i + 1
            k=k+1

        while j < len(separuhkanan):
            A[k] = separuhkanan[j]
            j = j + 1
            k=k+1
    #print("Menggabungkan",A)

def convert(arr, obj):
```

```python
def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(arr)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for x in Daftar:
    A.append(x.nim)

print("MERGE SORT")
mergeSort(A)
for x in convert(A, Daftar):
    print (x.nim)

##QuickSort
def partisi(A, awal, akhir):
    nilaipivot = A[awal]

    penandakiri = awal + 1
    penandakanan = akhir

    selesai = False
    while not selesai:

        while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1

        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
            penandakanan = penandakanan - 1

        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
            A[penandakiri] = A[penandakanan]
            A[penandakanan] = temp

    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah-1)
        quickSortBantu(A, titikBelah+1, akhir)

def quickSort(A):
    quickSortBantu (A, 0, len(A)-1)

def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(arr)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil
```

```python
        A.append(x.nim)

print("MERGE SORT")
mergeSort(A)
for x in convert(A, Daftar):
    print (x.nim)

##QuickSort
def partisi(A, awal, akhir):
    nilaipivot = A[awal]

    penandakiri = awal + 1
    penandakanan = akhir

    selesai = False
    while not selesai:

        while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1

        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
            penandakanan = penandakanan - 1

        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
            A[penandakiri] = A[penandakanan]
            A[penandakanan] = temp

    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah-1)
        quickSortBantu(A, titikBelah+1, akhir)

def quickSort(A):
    quickSortBantu (A, 0, len(A)-1)

def convert(arr, obj):
    hasil=[]
    for x in range (len(arr)):
        for i in range (len(arr)):
            if arr[x] == obj[i].nim:
                hasil.append(obj[i])
    return hasil

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for x in Daftar:
    A.append(x.nim)

print("QUICK SORT")
quickSort(A)
for x in convert(A, Daftar):
    print (x.nim)
```
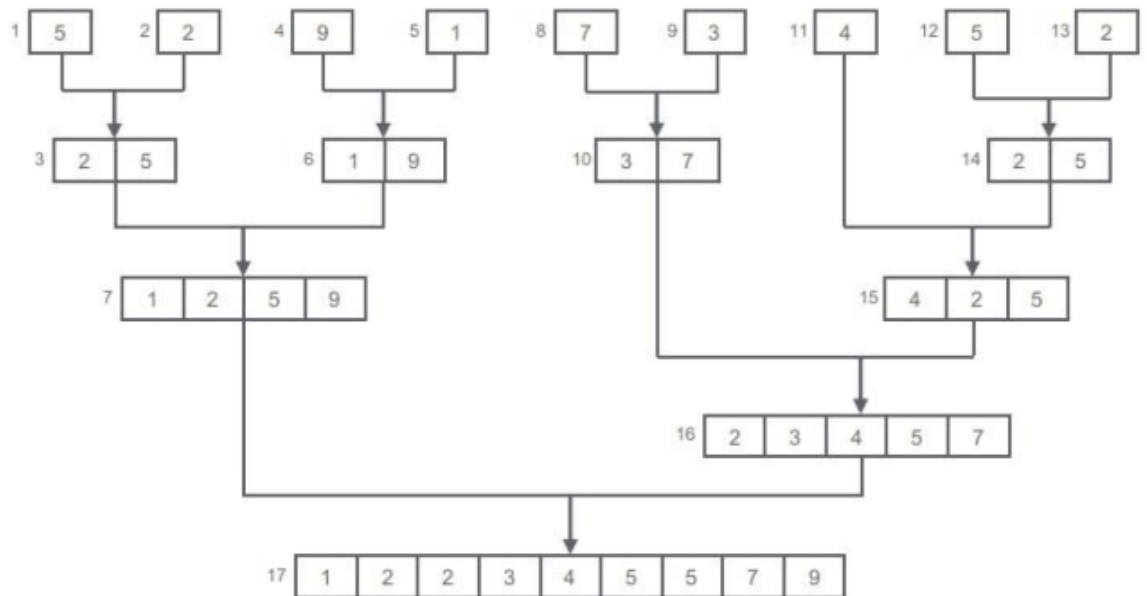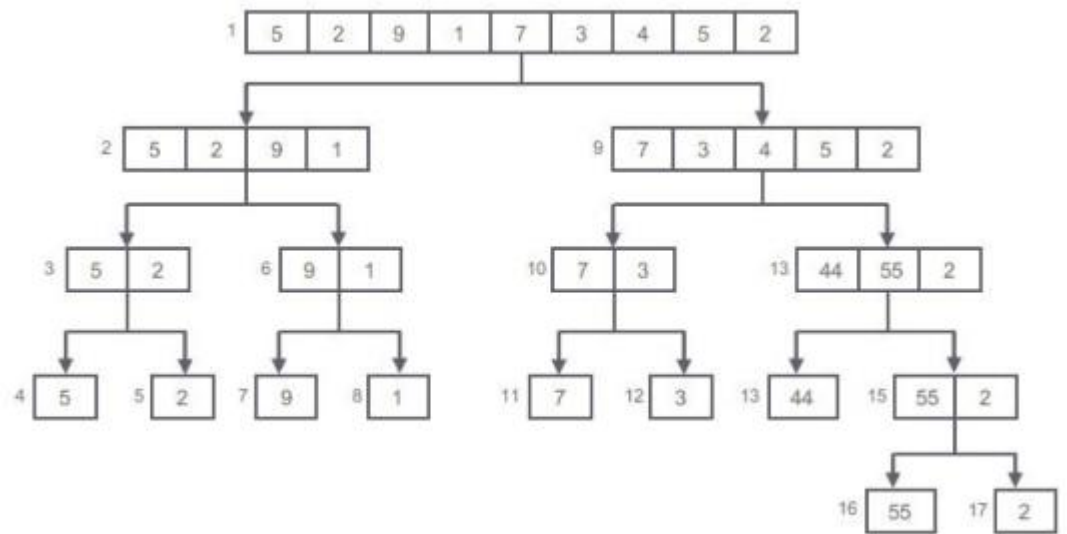
Setelah dirun hasilnya :

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: E:\Semester 4\prak algostruk\modul 6\L200180058_Algostruk_Modul6_Tugas
\No 1.py
MERGE SORT
2
4
5
10
13
18
23
29
31
51
64
QUICK SORT
2
4
5
10
13
18
23
29
31
51
64
>>>
```

Ln: 29  Col: 4

2. Tandai dan beri nomor urut eksekusi proses pada gambar hal 59



| 1 | 5 | 2 | 9 | 1 | 7 | 3 | 4 | 5 | 2 |

Merge sort tree diagram:

Level split (first tree):
- 1: [5 2 9 1 7 3 4 5 2]
- 2: [5 2 9 1]  |  9: [7 3 4 5 2]
- 3: [5 2]  6: [9 1]  |  10: [7 3]  13: [44 55 2]
- 4: [5]  5: [2]  7: [9]  8: [1]  |  11: [7]  12: [3]  13: [44]  15: [55 2]
- 16: [55]  17: [2]

Merge (second tree):
- 1: [5]  2: [2]  4: [9]  5: [1]  8: [7]  9: [3]  11: [4]  12: [5]  13: [2]
- 3: [2 5]  6: [1 9]  10: [3 7]  14: [2 5]
- 7: [1 2 5 9]  15: [4 2 5]
- 16: [2 3 4 5 7]
- 17: [1 2 2 3 4 5 5 7 9]

3. Menguji kecepatan mergeSort dan QuickSort

```python
from time import time as detak
from random import shuffle as kocok
import time

def swap(A, p, q):
    tmp = A[p]
    A[p] = A[q]
    A[q] = tmp

def cariPosisiYangTerkecil(A, dariSini, sampaiSini):
    posisiYangTerkecil = dariSini
    for i in range(dariSini+1, sampaiSini):
        if A[i] < A[posisiYangTerkecil]:
            posisiYangTerkecil = i
    return posisiYangTerkecil

def bubbleSort(S):
    n = len(S)
    for i in range (n-1):
        for j in range (n-i-1):
            if S[j] > S[j+1]:
                swap(S,j,j+1)
    return S

def selectionSort(S):
    n = len(S)
    for i in range(n-1):
        indexKecil = cariPosisiYangTerkecil(S, i, n)
        if indexKecil != i:
            swap(S, i, indexKecil)
    return S

def insertionSort(S):
    n = len(S)
    for i in range(1, n):
        nilai = S[i]
        pos = i
        while pos > 0 and nilai < S[pos -1]:
            S[pos] = S[pos-1]
            pos = pos - 1
        S[pos] = nilai
    return S

def mergeSort(A):
    #print("Membelah        ",A)
    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]

        mergeSort(separuhkiri)
        mergeSort(separuhkanan)

        i = 0;j=0;k=0
        while i < len(separuhkiri) and j < len(separuhkanan):
            if separuhkiri[i] < separuhkanan[j]:
                A[k] = separuhkiri[i]
                i = i + 1
            else:
                A[k] = separuhkanan[j]
                j = j + 1
            k=k+1
```

Ln: 1  Col: 0

```python
    #print("Menggabungkan",A)

def partisi(A, awal, akhir):
    nilaipivot = A[awal]

    penandakiri = awal + 1
    penandakanan = akhir

    selesai = False
    while not selesai:

        while penandakiri <= penandakanan and A[penandakiri] <= nilaipivot:
            penandakiri = penandakiri + 1

        while penandakanan >= penandakiri and A[penandakanan] >= nilaipivot:
            penandakanan = penandakanan - 1

        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
            A[penandakiri] = A[penandakanan]
            A[penandakanan] = temp

    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    return penandakanan

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah-1)
        quickSortBantu(A, titikBelah+1, akhir)

def quickSort(A):
    quickSortBantu(A, 0, len(A)-1)

daftar = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

print (bubbleSort(daftar))
print (selectionSort(daftar))
print (insertionSort(daftar))
mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mrg = k[:]
u_qck = k[:]

aw=detak();bubbleSort(u_bub);ak=detak();print("bubble: %g detik" %(ak-aw));
aw=detak();selectionSort(u_sel);ak=detak();print("selection: %g detik" %(ak-aw))
aw=detak();insertionSort(u_ins);ak=detak();print("insertion: %g detik" %(ak-aw))
aw=detak();mergeSort(u_mrg);ak=detak();print("merge: %g detik" %(ak-aw));
aw=detak();quickSort(u_qck);ak=detak();print("quick: %g detik" %(ak-aw));
```
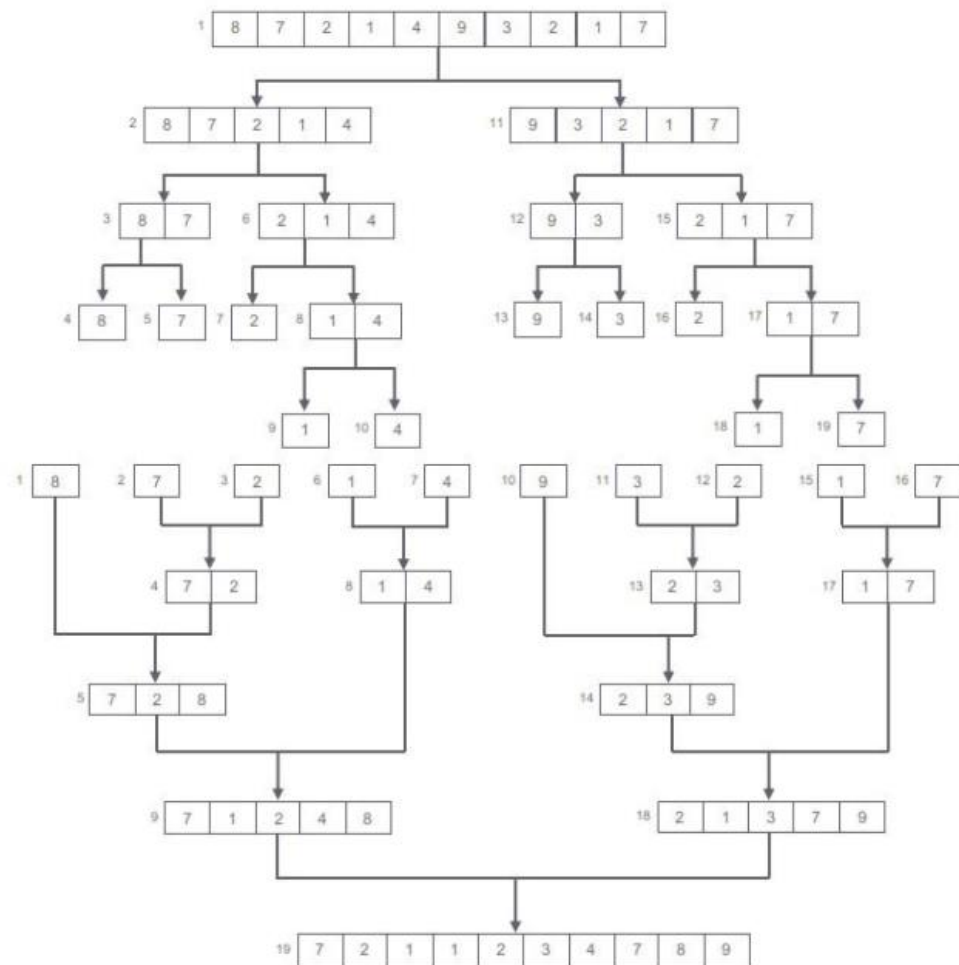
Ln: 1  Col: 0

Setelah dirun hasilnya:

```
bubble: 5.75773 detik
selection: 2.43541 detik
insertion: 2.8584 detik
merge: 0.0389307 detik
quick: 0.0259638 detik
```

4. Diberikan list L = [80,7,24,16,43,91,35,2,19,72], gambarkan trace pengurutan untuk agoritma

5. Meningkatkan efesiensi program mergeSort dengan tidak memakai operator slice dan lalu mem-pass index awal dan index akhir Bersama list-nya saat memanggil mergeSort secara rekursif

```python
class MhsTIF():
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kota = kota
        self.us = us

    def __str__(self):
        s = self.nama +', NIM '+str(self.nim)\
            +'. Tinggal di '+ self.kota \
            +'. Uang saku Rp. '+ str(self.us)\
            +' tiap bulannya.'
        return s

    def ambilNama(self):
        return self.nama
    def ambilNim(self):
        return self.nim
    def ambilUangSaku(self):
        return self.us

c0 = MhsTIF("Ika", 10, "Sukoharjo", 240000)
c1 = MhsTIF("Budi", 51, "Sragen", 230000)
c2 = MhsTIF("Ahmad", 2, "Surakarta", 250000)
c3 = MhsTIF("Chandra", 18, "Surakarta", 235000)
c4 = MhsTIF("Eka", 4, "Boyolali", 240000)
c5 = MhsTIF("Fandi", 31, "Salatiga", 250000)
c6 = MhsTIF("Deni", 13, "Klaten", 245000)
c7 = MhsTIF("Galuh", 5, "Wonogiri", 245000)
c8 = MhsTIF("Janto", 23, "Klaten", 245000)
c9 = MhsTIF("Hasan", 64, "Karanganyar", 270000)
c10 = MhsTIF("Khalid", 29, "Purwodadi", 265000)

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]

def cetak(A):
    for i in A:
        print (i)

def mergeSort2(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergeSort2(A, awal, mid)
        mergeSort2(A, mid+1, akhir)

    a, f, l = 0, awal, mid+1
    tmp = [None] * (akhir - awal + 1)
    while f <= mid and l <= akhir:
        if A[f].ambilUangSaku() < A[l].ambilUangSaku():
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1

    if f <= mid:
        tmp[a:] = A[f:mid+1]

    if l <= akhir:
        tmp[a:] = A[l:akhir+1]

    a = 0
```

Setelah dirun hasilnya:

```
>>>
 RESTART: E:\Semester 4\prak algostruk\modul 6\L200180058_Algostruk_Modul6_Tug
\No 5.py
Budi, NIM 51. Tinggal di Sragen. Uang saku Rp. 230000 tiap bulannya.
Chandra, NIM 18. Tinggal di Surakarta. Uang saku Rp. 235000 tiap bulannya.
Eka, NIM 4. Tinggal di Boyolali. Uang saku Rp. 240000 tiap bulannya.
Ika, NIM 10. Tinggal di Sukoharjo. Uang saku Rp. 240000 tiap bulannya.
Janto, NIM 23. Tinggal di Klaten. Uang saku Rp. 245000 tiap bulannya.
Galuh, NIM 5. Tinggal di Wonogiri. Uang saku Rp. 245000 tiap bulannya.
Deni, NIM 13. Tinggal di Klaten. Uang saku Rp. 245000 tiap bulannya.
Fandi, NIM 31. Tinggal di Salatiga. Uang saku Rp. 250000 tiap bulannya.
Ahmad, NIM 2. Tinggal di Surakarta. Uang saku Rp. 250000 tiap bulannya.
Khalid, NIM 29. Tinggal di Purwodadi. Uang saku Rp. 265000 tiap bulannya.
Hasan, NIM 64. Tinggal di Karanganyar. Uang saku Rp. 270000 tiap bulannya.
>>>
```

6. Meningkatkan efesiensi quicksort dengan memakai metode median dari tiga untuk memilih pivotnya

```python
class MhsTIF():
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kota = kota
        self.us = us

    def __str__(self):
        s = self.nama +', NIM '+str(self.nim)\
            +'. Tinggal di '+ self.kota \
            +'. Uang saku Rp. '+ str(self.us)\
            +' tiap bulannya.'
        return s

    def ambilNama(self):
        return self.nama
    def ambilNim(self):
        return self.nim
    def ambilUangSaku(self):
        return self.us

c0 = MhsTIF("Ika", 10, "Sukoharjo", 240000)
c1 = MhsTIF("Budi", 51, "Sragen", 230000)
c2 = MhsTIF("Ahmad", 2, "Surakarta", 250000)
c3 = MhsTIF("Chandra", 18, "Surakarta", 235000)
c4 = MhsTIF("Eka", 4, "Boyolali", 240000)
c5 = MhsTIF("Fandi", 31, "Salatiga", 250000)
c6 = MhsTIF("Deni", 13, "Klaten", 245000)
c7 = MhsTIF("Galuh", 5, "Wonogiri", 245000)
c8 = MhsTIF("Janto", 23, "Klaten", 245000)
c9 = MhsTIF("Hasan", 64, "Karanganyar", 270000)
c10 = MhsTIF("Khalid", 29, "Purwodadi", 265000)

Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
A = []
for i in Daftar:
    A.append(i.nama)

def cetak():
    for i in A:
        print(i)

def quickSort(arr):
    kurang = []
    pivotList = []
    lebih = []
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        for i in arr:
            if i < pivot:
                kurang.append(i)
            elif i > pivot:
                lebih.append(i)
            else:
                pivotList.append(i)
        kurang = quickSort(kurang)
        lebih = quickSort(lebih)
        return kurang + pivotList + lebih

print("Sebelum diurutkan")
cetak()
```

Ln: 1  Col: 0

Sesudah dirun hasilnya:

```
 RESTART: E:\Semester 4\prak algostruk\modul 6\L200180058_Algostruk_Modul6_
\No 6.py
Sebelum diurutkan
Ika
Budi
Ahmad
Chandra
Eka
Fandi
Deni
Galuh
Janto
Hasan
Khalid

Setelah diurutkan
Ika
Budi
Ahmad
Chandra
Eka
Fandi
Deni
Galuh
Janto
Hasan
Khalid
>>> |
```

7.

```
merge: 0.0361111 detik
quick: 0.0224726 detik
merge New: 0.0458734 detik
quick New: 0.0175741 detik
>>>
```

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


list1 = LinkedList()
list1.appendSorted(14)
list1.appendSorted(13)
list1.appendSorted(5)
list1.appendSorted(19)
list1.appendSorted(9)
```

Jika dirun maka hasilnya :

```
>>>
 RESTART: E:\Semester 4\prak algostruk\modul 6\L200180058_Algostruk_
\No 8.py
List 1 :
5
9
13
14
19
List 2 :
1
15
30
Merged List :
1
5
9
13
14
15
19
30
>>> |
```