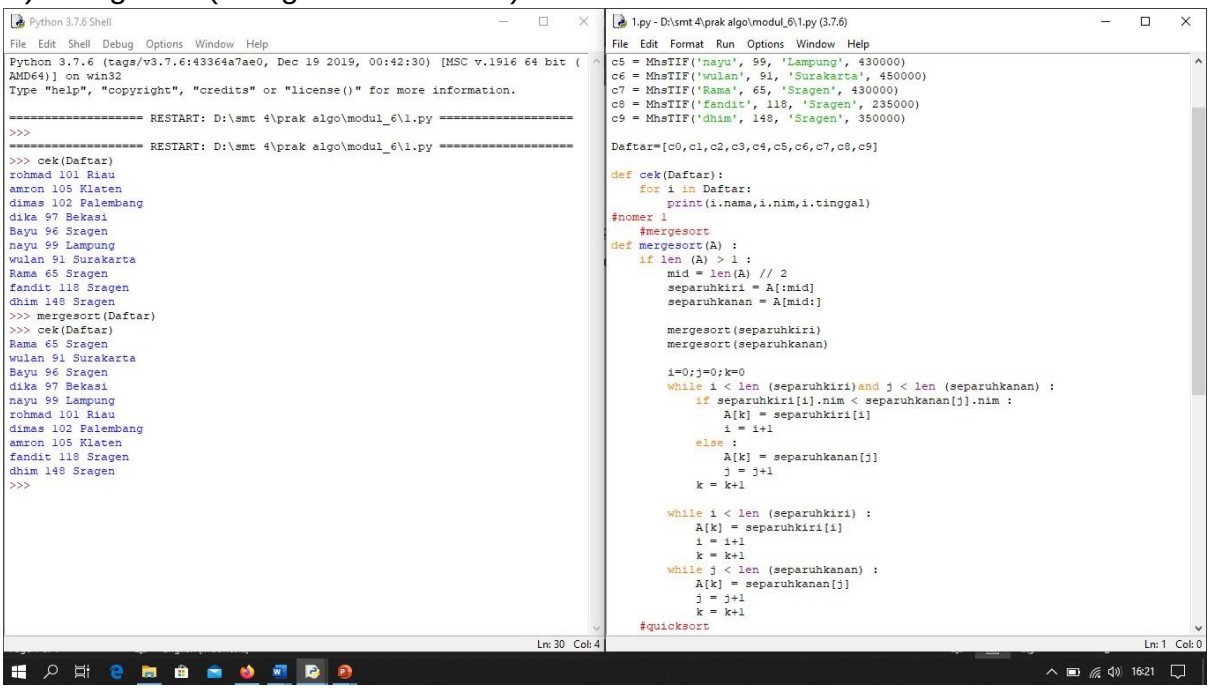
Nama : Chandika Aulia
Nim : L200180097
Kelas : D

Prak-ASD

Modul 6

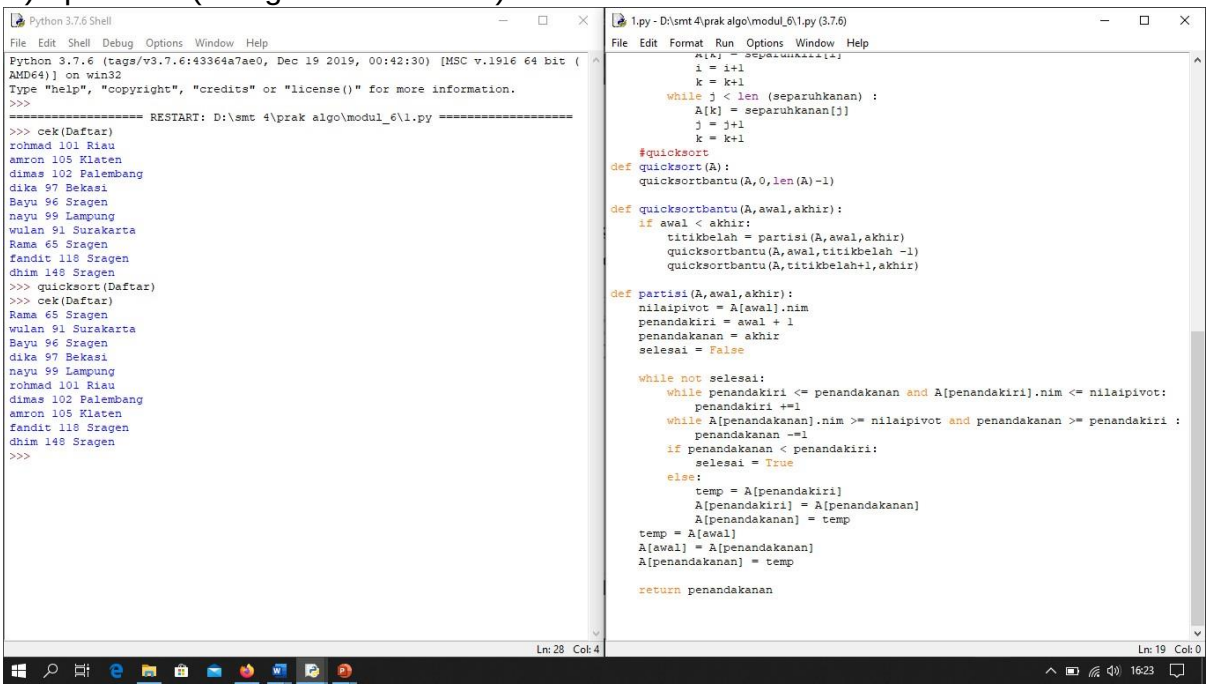
1. A). Mergesort (mengurutkan MhsTif)



```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

================== RESTART: D:\smt 4\prak algo\modul_6\1.py ==================
>>>
================== RESTART: D:\smt 4\prak algo\modul_6\1.py ==================
>>> cek(Daftar)
rohmad 101 Riau
amron 105 Klaten
dimas 102 Palembang
dika 97 Bekasi
Bayu 96 Sragen
nayu 99 Lampung
wulan 91 Surakarta
Rama 65 Sragen
fandit 118 Sragen
dhim 148 Sragen
>>> mergesort(Daftar)
>>> cek(Daftar)
Rama 65 Sragen
wulan 91 Surakarta
Bayu 96 Sragen
dika 97 Bekasi
nayu 99 Lampung
rohmad 101 Riau
dimas 102 Palembang
amron 105 Klaten
fandit 118 Sragen
dhim 148 Sragen
>>>
```

```
c5 = MhsTIF('nayu', 99, 'Lampung', 430000)
c6 = MhsTIF('wulan', 91, 'Surakarta', 450000)
c7 = MhsTIF('Rama', 65, 'Sragen', 430000)
c8 = MhsTIF('fandit', 118, 'Sragen', 235000)
c9 = MhsTIF('dhim', 148, 'Sragen', 350000)

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

def cek(Daftar):
    for i in Daftar:
        print(i.nama,i.nim,i.tinggal)
#nomer 1
    #mergesort
def mergesort(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].nim < separuhkanan[j].nim :
                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
    #quicksort
```

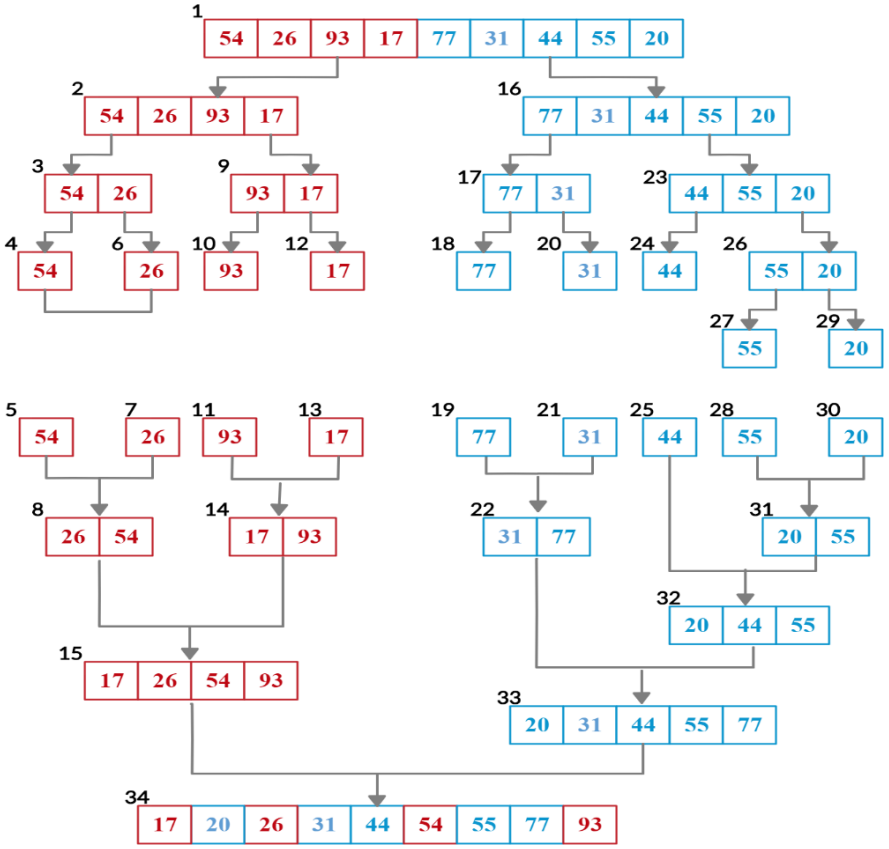
B). quicksort (Mengurutkan MhsTif)



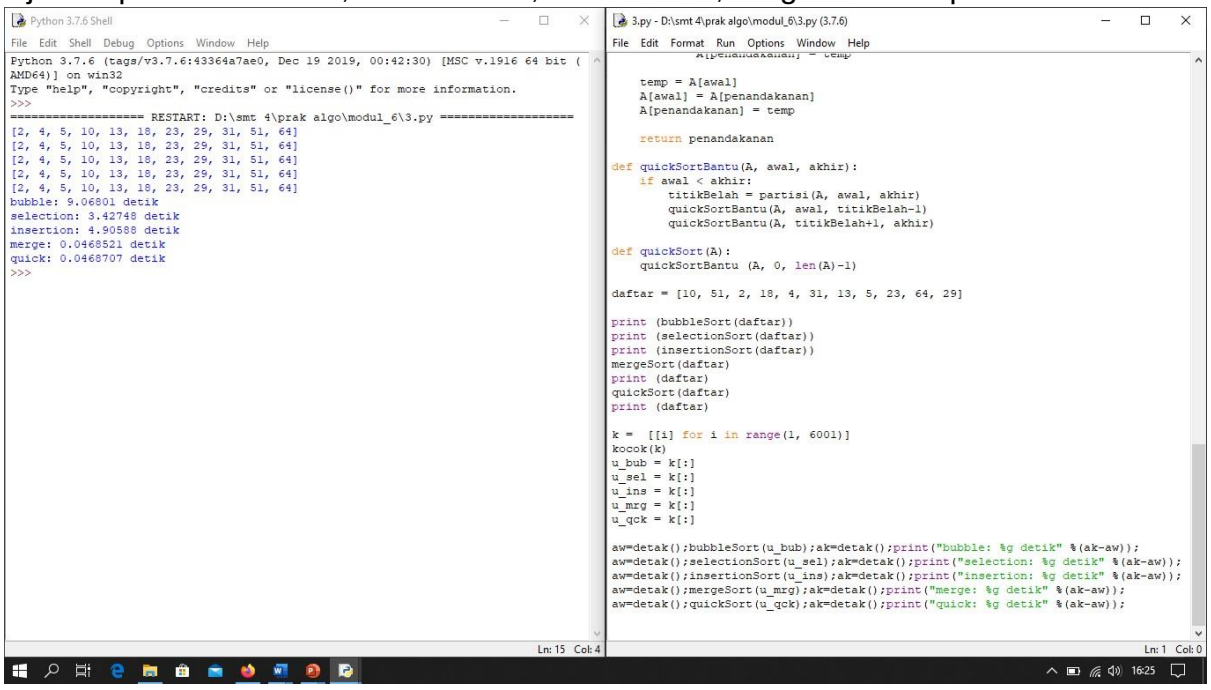
```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\smt 4\prak algo\modul_6\1.py ==================
>>> cek(Daftar)
rohmad 101 Riau
amron 105 Klaten
dimas 102 Palembang
dika 97 Bekasi
Bayu 96 Sragen
nayu 99 Lampung
wulan 91 Surakarta
Rama 65 Sragen
fandit 118 Sragen
dhim 148 Sragen
>>> quicksort(Daftar)
>>> cek(Daftar)
Rama 65 Sragen
wulan 91 Surakarta
Bayu 96 Sragen
dika 97 Bekasi
nayu 99 Lampung
rohmad 101 Riau
dimas 102 Palembang
amron 105 Klaten
fandit 118 Sragen
dhim 148 Sragen
>>>
```

```
            i = i+1
            k = k+1
        while j < len (separuhkanan) :
            A[k] = separuhkanan[j]
            j = j+1
            k = k+1
    #quicksort
def quicksort(A):
    quicksortbantu(A,0,len(A)-1)

def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

def partisi(A,awal,akhir):
    nilaipivot = A[awal].nim
    penandakiri = awal + 1
    penandakanan = akhir
    selesai = False

    while not selesai:
        while penandakiri <= penandakanan and A[penandakiri].nim <= nilaipivot:
            penandakiri +=1
        while A[penandakanan].nim >= nilaipivot and penandakanan >= penandakiri :
            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
```

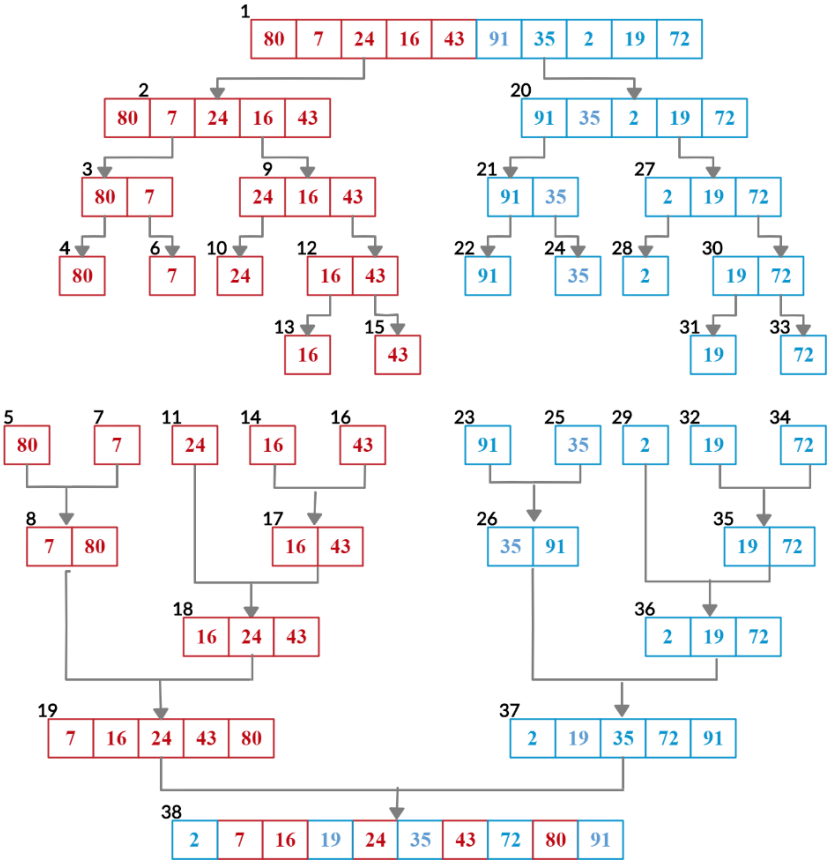
2. Beri nomer urut eksekusi proses gambar 6.1 dan 6.2 mengacu pada output di halaman 59



3. Uji kecepatan bublesort,selectionsort,insertionsort,mergesort dan quicksort



4. A. diberikan List = [80,7,24,16,43,91,35,2,19,72] ,gambarlah trace pengurutan algoritmanya (Merge sort)



4 B. diberikan List = [80,7,24,16,43,91,35,2,19,72] ,gambarlah trace pengurutan algoritmanya (quickSort)

80	7	24	16	43	91	35	2	19	72

pivot

80	7	24	16	43	91	35	2	19	72

Low High

pivot

72	7	24	16	43	91	35	2	19	80

Low High

pivot

72	7	24	16	43	91	35	2	19	80

Low High

pivot

72	7	24	16	43	80	35	2	19	91
					Low				High

pivot

72	7	24	16	43	19	35	2	80	91
					Low			High	

pivot

72	7	24	16	43	19	35	2	80	91
Low							High		

pivot

2	7	24	16	43	19	35	72	80	91
Low							High		

pivot

2	7	24	16	43	19	35	72	80	91
Low						High			

pivot

2	7	24	16	43	19	35	72	80	91
	Low					High			

pivot

2	7	24	16	43	19	35	72	80	91
		Low				High			

pivot

2	7	24	16	43	19	35	72	80	91
		Low			High				

pivot

2	7	19	16	43	24	35	72	80	91
		Low			High				

pivot

2	7	19	16	43	24	35	72	80	91
				Low	High				

pivot

2	7	19	16	24	43	35	72	80	91
				Low	High				

pivot

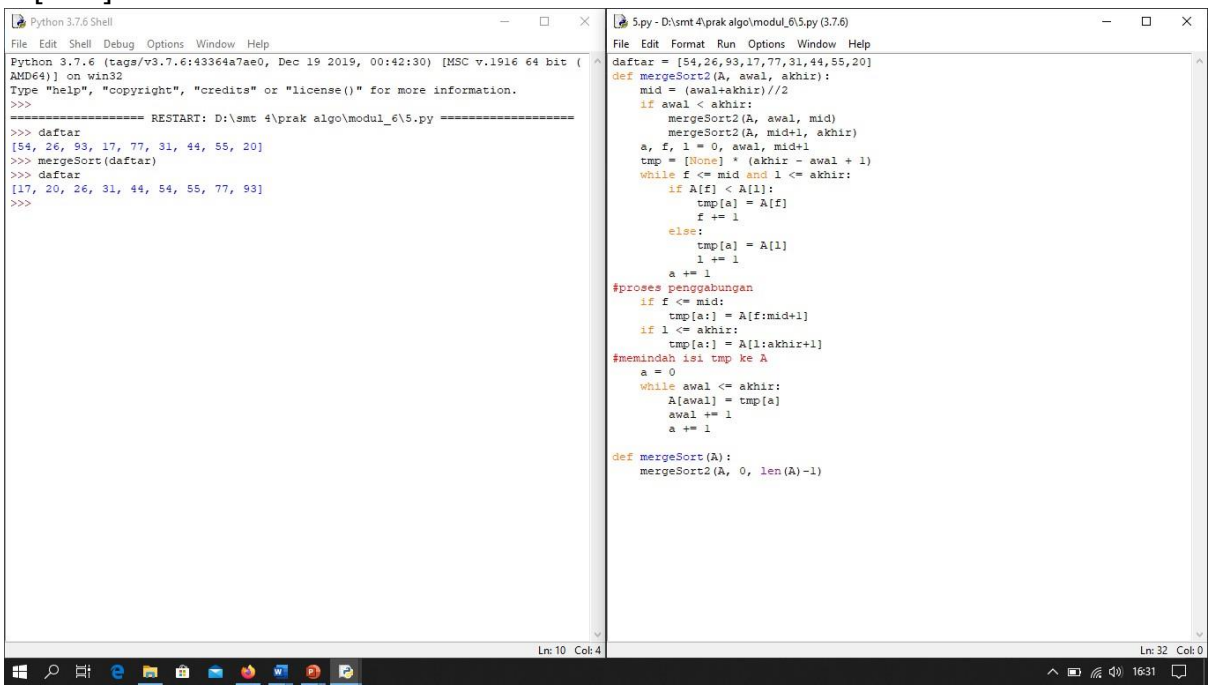
2	7	19	16	24	43	35	72	80	91
		Low	High						

pivot

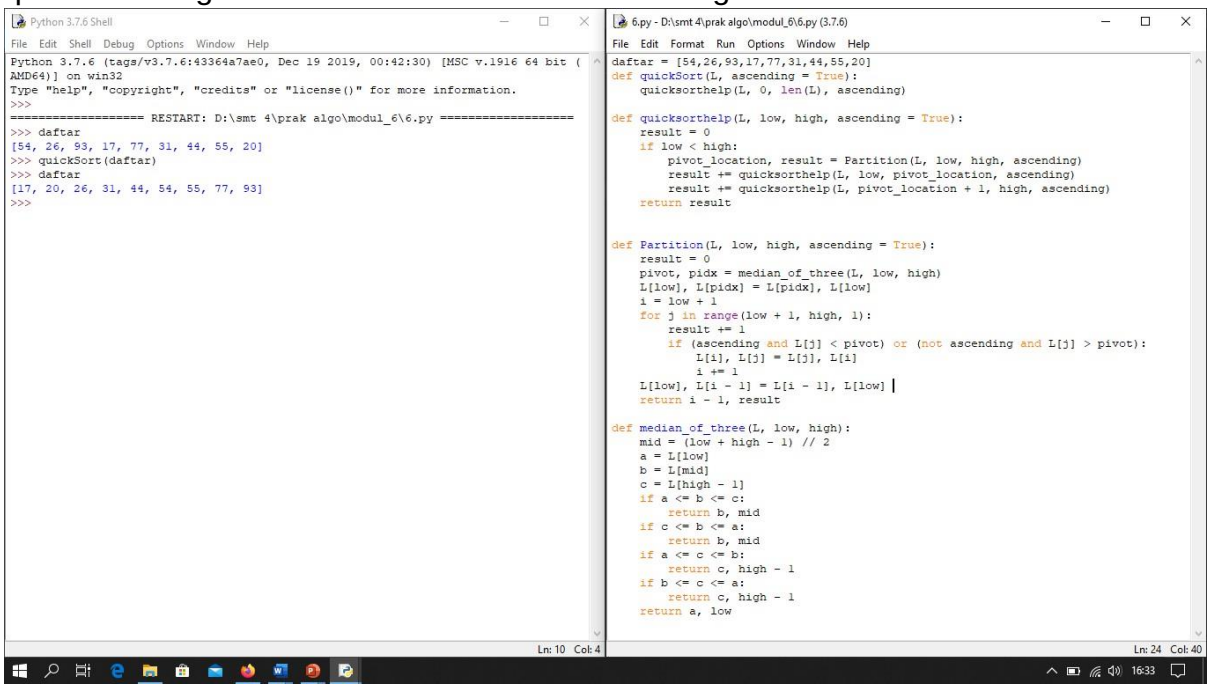
2	7	16	19	24	35	43	72	80	91
				Low	High				

2	7	16	19	24	35	43	72	80	91

5. Tingkatkan efisien mergesort dengan tidak memakai operator A[:mid] dan A [mid:]



6. quicksort dengan memakai metode median-dari-tiga



7. uji kecepatan program nmr 5, 6 ,mergesort (awal) dan quicksort (akhir)

Python 3.7.6 Shell

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\smt 4\prak algo\modul_6\7.py ==================
mergesort: 0.078126 detik
mergesort terbaru: 0.0781353 detik
quicksort: 0.0468318 detik
quicksort terbaru: 0.0468888 detik
>>>
```

7.py - D:\smt 4\prak algo\modul_6\7.py (3.7.6)

```
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i - 1] = L[i - 1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low + high - 1) // 2
    a = L[low]
    b = L[mid]
    c = L[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 = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]
from time import time as detak
from random import shuffle as kocok
import time

k =  [[i] for i in range(1, 6001)]
kocok(k)
u_mer = k[:]
u_mer5 = k[:]
u_qui = k[:]
u_qui6 = k[:]

aw=detak();mergesort(u_mer);ak=detak();print("mergesort: %g detik" %(ak-aw));
aw=detak();mergesort_5(u_mer5);ak=detak();print("mergesort terbaru: %g detik" %(a
aw=detak();quicksort(u_qui);ak=detak();print("quicksort: %g detik" %(ak-aw));
aw=detak();quicksort_6(u_qui6);ak=detak();print("quicksort terbaru: %g detik" %(a
```

8. versi linked list mergesort

Python 3.7.6 Shell

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\smt 4\prak algo\modul_6\8.py ==================
List 1 :
16
17
33
48
92
List 2 :
10
18
23
Merged List :
10
16
17
18
23
33
48
92
>>>
```

8.py - D:\smt 4\prak algo\modul_6\8.py (3.7.6)

```
        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 = Linked()
list1.appendSorted(48)
list1.appendSorted(92)
list1.appendSorted(33)
list1.appendSorted(16)
list1.appendSorted(17)

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

list2 = Linked()
list2.appendSorted(23)
list2.appendSorted(10)
list2.appendSorted(18)

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

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

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