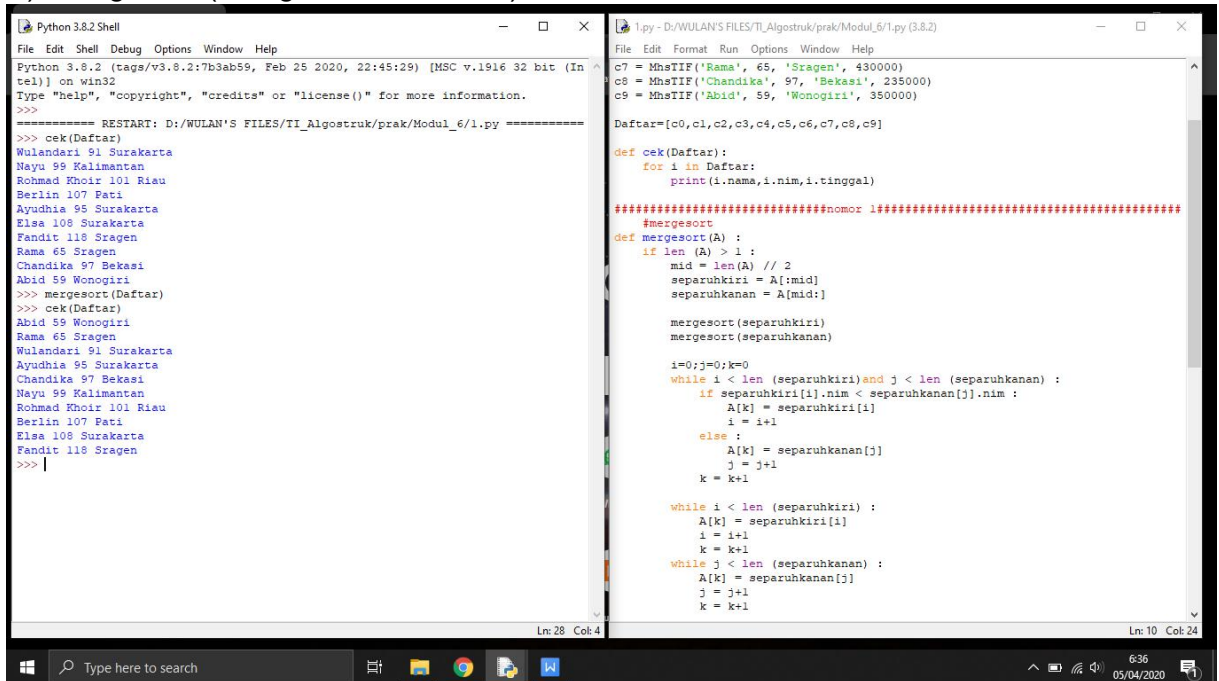


Nama : Wulandari Ratna Kartika Jayawardani
Nim : L200180091
Kelas : D

Modul 6

1. A). Mergesort (Mengurutkan MhsTif)



The screenshot shows two windows from a Windows 10 desktop. The left window is a Python 3.8.2 Shell with the following code:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/II_Algostruk/prak/Modul_6/1.py =====
>>> cek(Daftar)
Wulandari 91 Surakarta
Nayu 99 Kalimantan
Rohmad Khoir 101 Riau
Berlin 107 Pati
Ayudhia 95 Surakarta
Elaa 108 Surakarta
Fandit 118 Sragen
Rama 65 Sragen
Chandika 97 Bekasi
Abid 59 Wonogiri
>>> mergesort(Daftar)
>>> cek(Daftar)
Abid 59 Wonogiri
Rama 65 Sragen
Wulandari 91 Surakarta
Ayudhia 95 Surakarta
Chandika 97 Bekasi
Nayu 99 Kalimantan
Rohmad Khoir 101 Riau
Berlin 107 Pati
Elaa 108 Surakarta
Fandit 118 Sragen
>>> |
```

The right window is a Python script editor showing the implementation of the Mergesort algorithm:

```
1.py - D:/WULAN'S FILES/II_Algostruk/prak/Modul_6/1.py (3.8.2)
File Edit Format Run Options Window Help

c7 = MhsTIF('Rama', 65, 'Sragen', 430000)
c8 = MhsTIF('Chandika', 97, 'Bekasi', 235000)
c9 = MhsTIF('Abid', 59, 'Wonogiri', 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)

##### nomor 1 #####

def mergesort(A):
    if len(A) > 1:
        mid = len(A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]

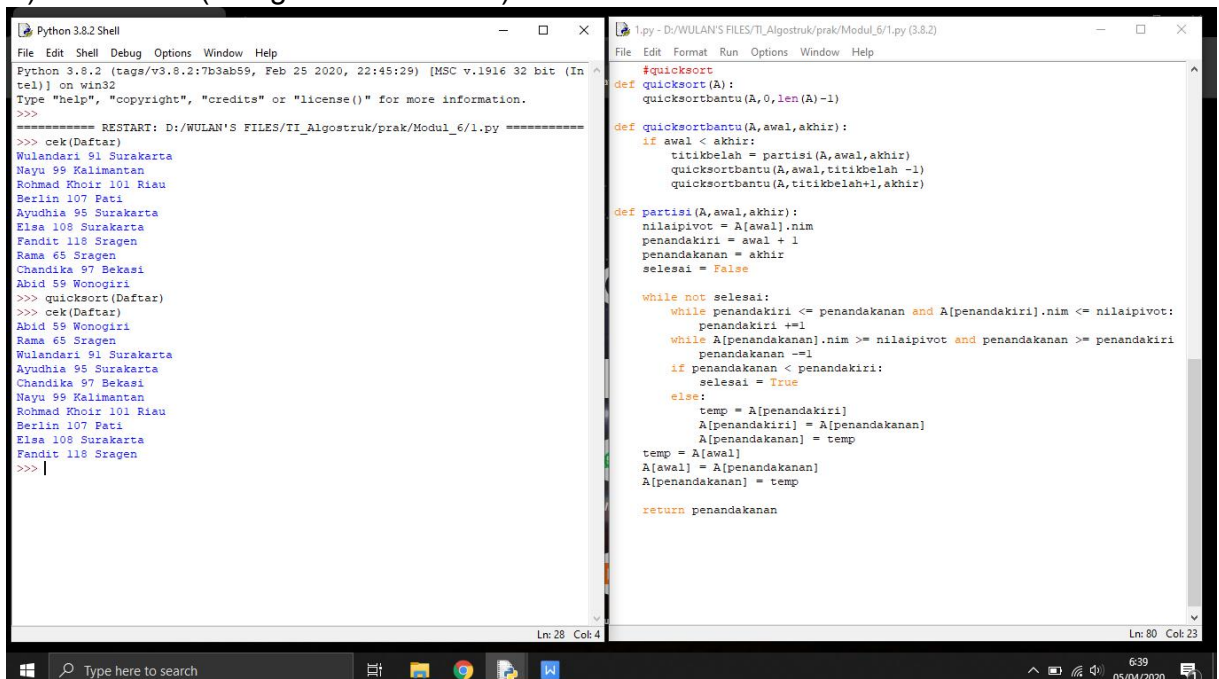
        mergesort(separuhkiri)
        mergesort(separuhkanan)

    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
```

B). Quicksort (Mengurutkan MhsTif)



The screenshot shows two windows from a Windows 10 desktop. The left window is a Python 3.8.2 Shell with the following code:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/II_Algostruk/prak/Modul_6/1.py =====
>>> cek(Daftar)
Wulandari 91 Surakarta
Nayu 99 Kalimantan
Rohmad Khoir 101 Riau
Berlin 107 Pati
Ayudhia 95 Surakarta
Elaa 108 Surakarta
Fandit 118 Sragen
Rama 65 Sragen
Chandika 97 Bekasi
Abid 59 Wonogiri
>>> quicksort(Daftar)
>>> cek(Daftar)
Abid 59 Wonogiri
Rama 65 Sragen
Wulandari 91 Surakarta
Ayudhia 95 Surakarta
Chandika 97 Bekasi
Nayu 99 Kalimantan
Rohmad Khoir 101 Riau
Berlin 107 Pati
Elaa 108 Surakarta
Fandit 118 Sragen
>>> |
```

The right window is a Python script editor showing the implementation of the Quicksort algorithm:

```
1.py - D:/WULAN'S FILES/II_Algostruk/prak/Modul_6/1.py (3.8.2)
File Edit Format Run Options Window Help

#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)

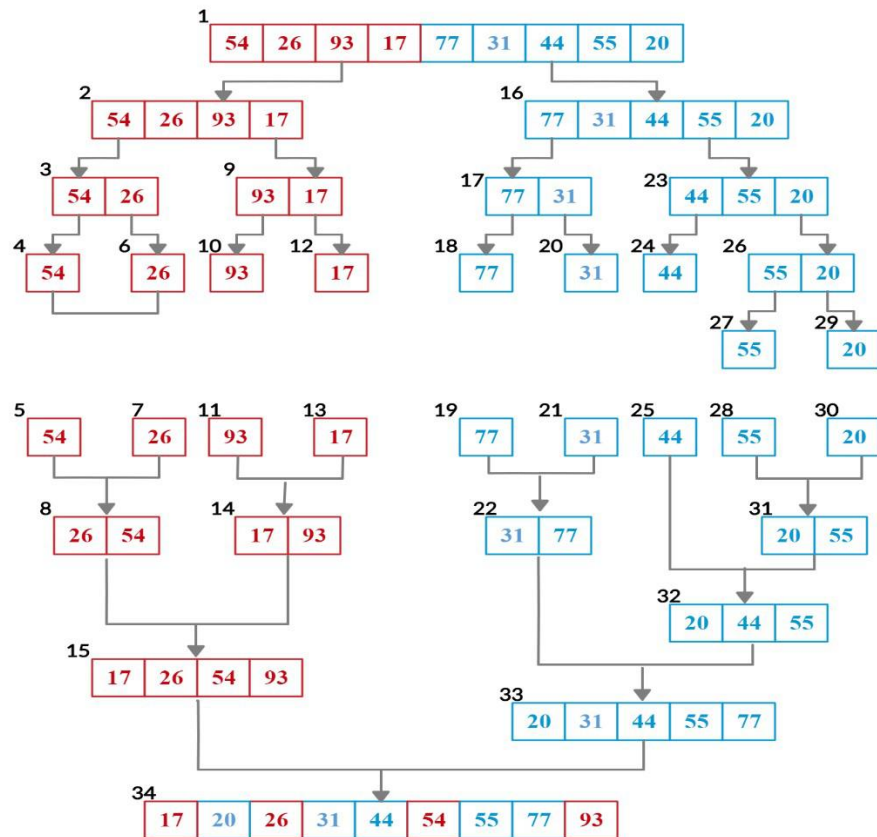
def partisi(A,awal,akhir):
    nilaipivot = A[awal].nim
    penandakiri = awal + 1
    penandakanan = akhir
    selesai = False

    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 bubblesort,selectionsort,insertionsort,mergesort dan quicksort

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/TI_Algostruk/prak/Modul_6/3.py =====
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
[2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
bubble: 14.0576 detik
selection: 6.65018 detik
insertion: 6.54689 detik
merge: 0.0762239 detik
quick: 0.0802348 detik
>>>

3.py - D:/WULAN'S FILES/TI_Algostruk/prak/Modul_6/3.py (3.8.2)
File Edit Format Run Options Window Help
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)

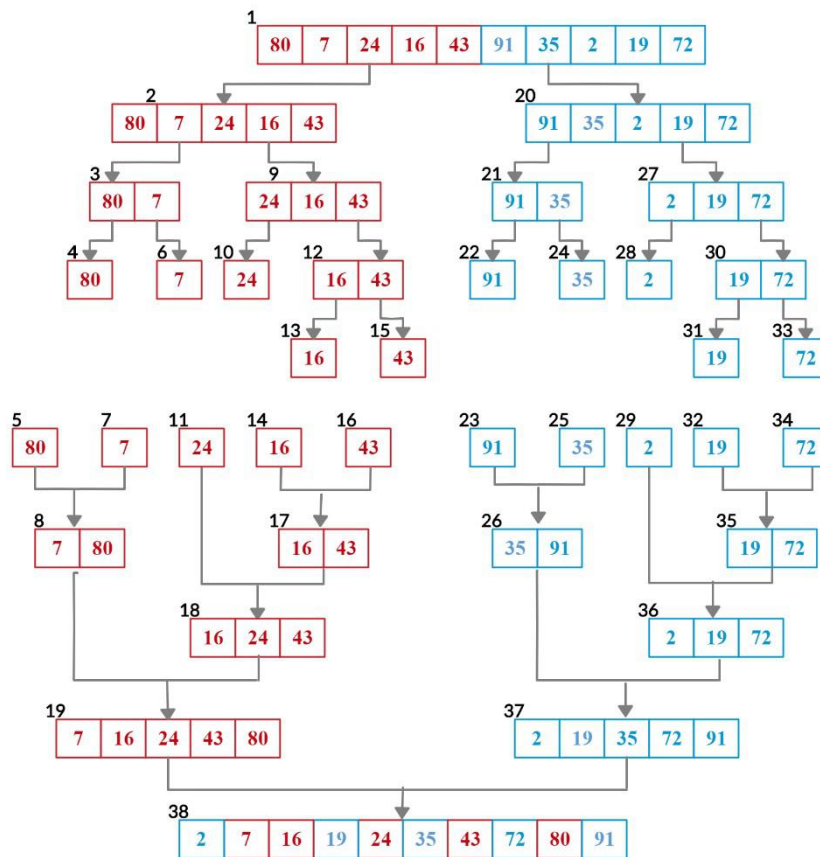
daftar = [10, 51, 2, 18, 4, 31, 13, 5, 23, 64, 29]

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[:]

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

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				

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

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

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

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

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

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

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

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

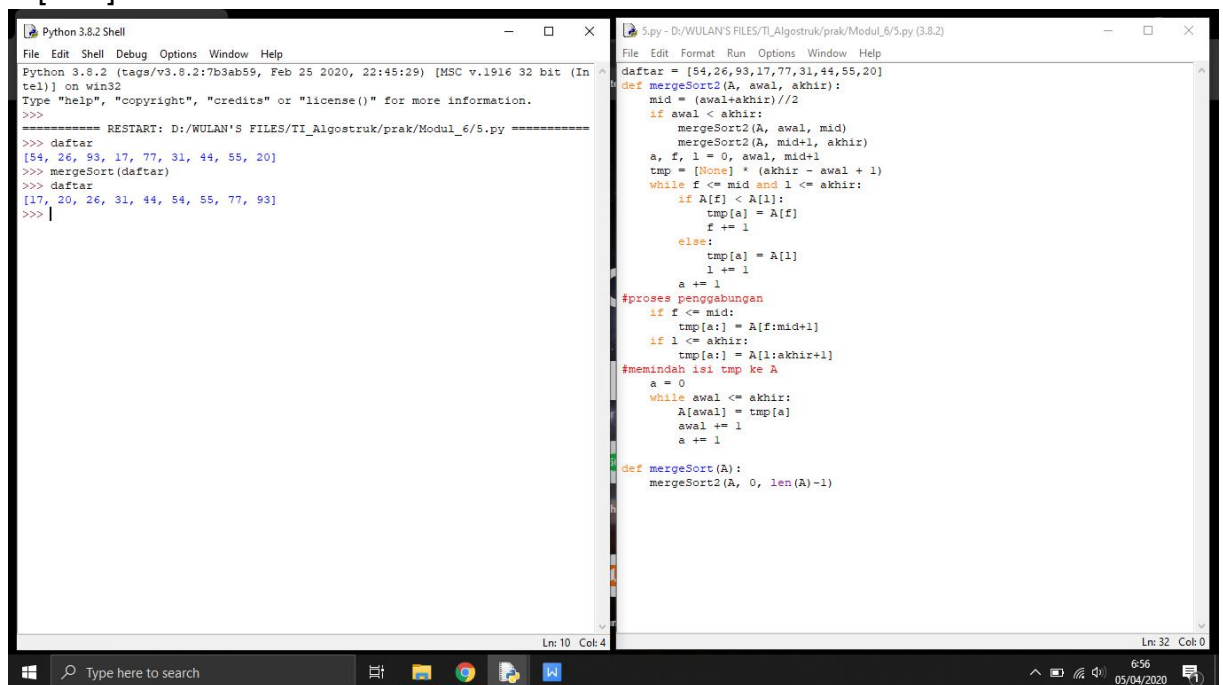
2	7	19	16	24	43	35	72	80	91
Low			High						

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 $A[mid:]$



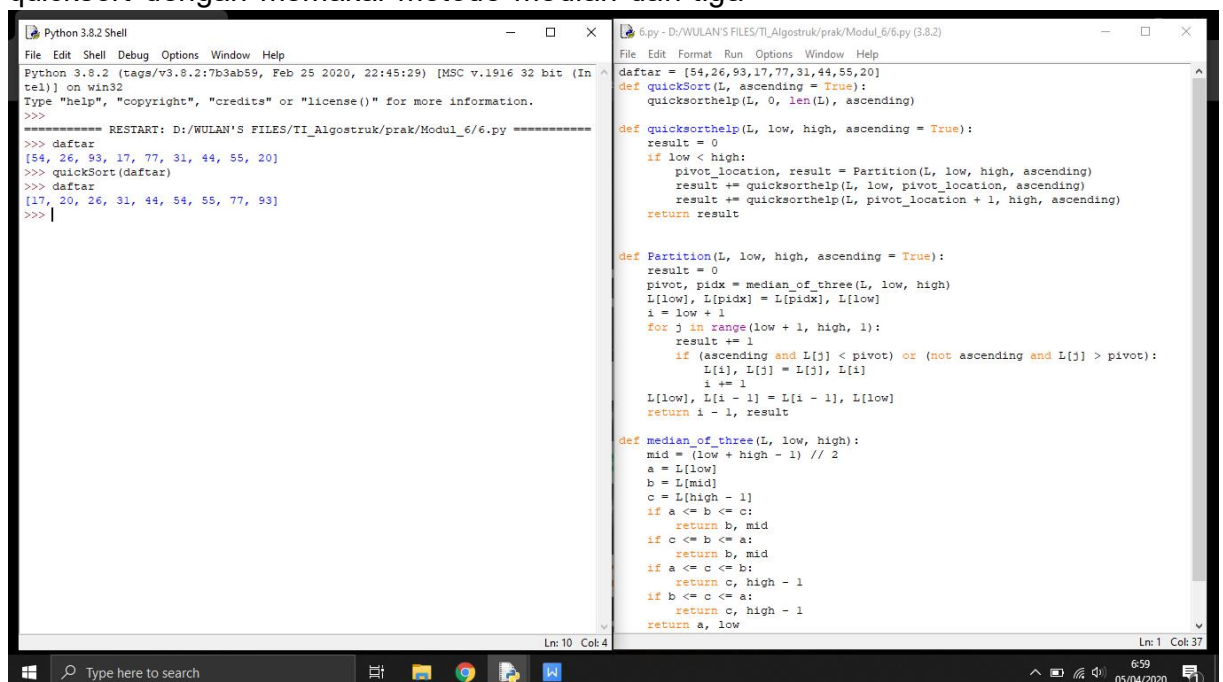
The screenshot shows two windows from a Windows desktop. The left window is a 'Python 3.8.2 Shell' with the following code and output:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/5.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> mergeSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> |
```

The right window is a Python 3.8.2 file editor showing the implementation of mergeSort2 and mergeSort:

```
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] < A[l]:
            tmp[a] = A[f]
            f += 1
        else:
            tmp[a] = A[l]
            l += 1
        a += 1
    #proses penggabungan
    if f <= mid:
        tmp[a:] = A[f:mid+1]
    if l <= akhir:
        tmp[a:] = A[l:akhir+1]
    #memindah isi tmp ke A
    a = 0
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1
    a = 1
def mergeSort(A):
    mergeSort2(A, 0, len(A)-1)
```

6. quicksort dengan memakai metode median-dari-tiga



The screenshot shows two windows from a Windows desktop. The left window is a 'Python 3.8.2 Shell' with the following code and output:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/6.py =====
>>> daftar
[54, 26, 93, 17, 77, 31, 44, 55, 20]
>>> quickSort(daftar)
>>> daftar
[17, 20, 26, 31, 44, 54, 55, 77, 93]
>>> |
```

The right window is a Python 3.8.2 file editor showing the implementation of quickSort, quickSorthelp, Partition, and median_of_three:

```
def quickSort(L, ascending = True):
    quickSorthelp(L, 0, len(L) - 1, ascending)

def quickSorthelp(L, low, high, ascending = True):
    result = 0
    if low < high:
        pivot_location, result = Partition(L, low, high, ascending)
        result += quickSorthelp(L, low, pivot_location, ascending)
        result += quickSorthelp(L, pivot_location + 1, high, ascending)
    return result

def Partition(L, low, high, ascending = True):
    result = 0
    pivot_idx = median_of_three(L, low, high)
    L[low], L[pivot_idx] = L[pivot_idx], L[low]
    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
```

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

The screenshot shows two windows from a Windows desktop. The left window is a Python 3.8.2 Shell with the following output:

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/7.py =====
mergesort : 0.0967321 detik
mergesort terbaru : 0.119668 detik
quicksort : 0.0623477 detik
quicksort terbaru : 0.0874674 detik
>>> |
```

The right window shows a Python script named 7.py with the following code:

```
7.py - D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/7.py (3.8.2)
File Edit Format Run Options Window Help

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

8. versi linked list mergesort

The screenshot shows two windows from a Windows desktop. The left window is a Python 3.8.2 Shell with the following output:

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/8.py =====
List 1 :
16
17
33
48
92
List 2 :
10
18
23
Mergesort Linked list :
10
16
17
18
23
33
48
92
>>> |
```

The right window shows a Python script named 8.py with the following code:

```
8.py - D:/WULAN'S FILES/TL_Algostruk/prak/Modul_6/8.py (3.8.2)
File Edit Format Run Options Window Help

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 = 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)
```