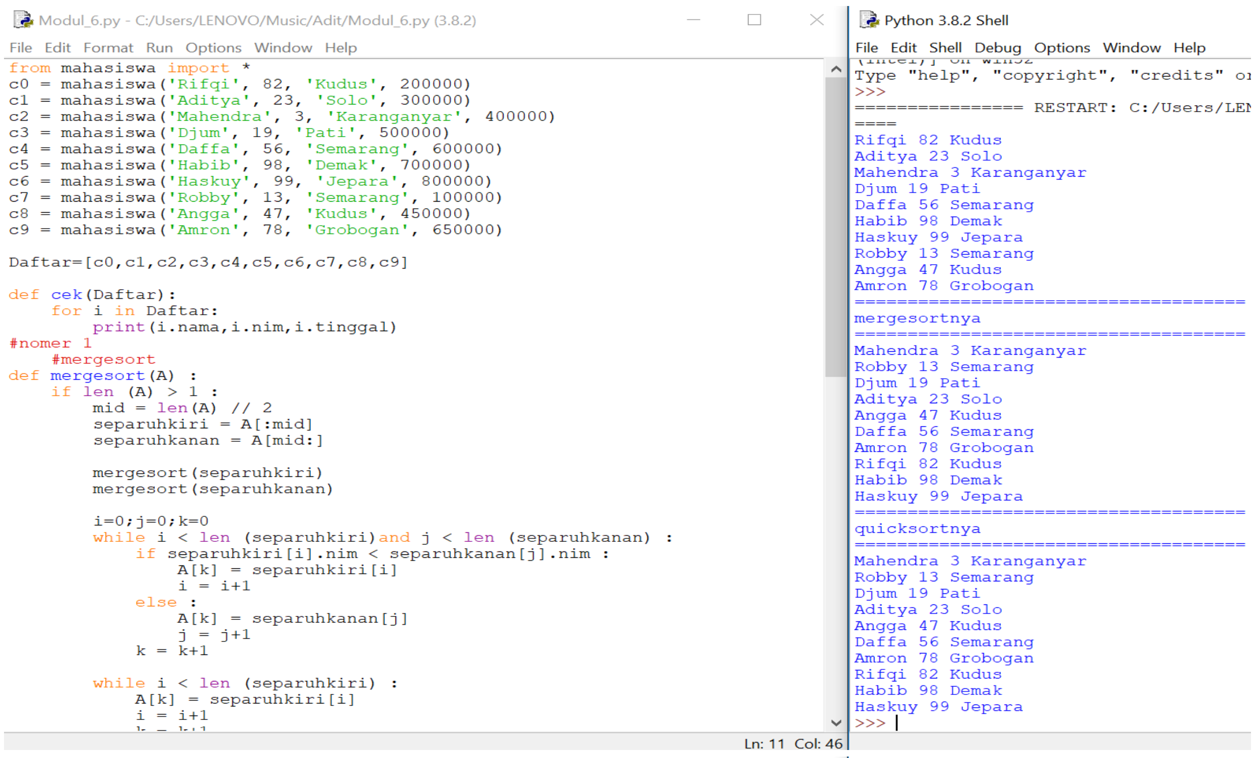
Nama : Rifqi Aditya Mahendra

NIM : L200180083

Kelas : D

MODUL 6

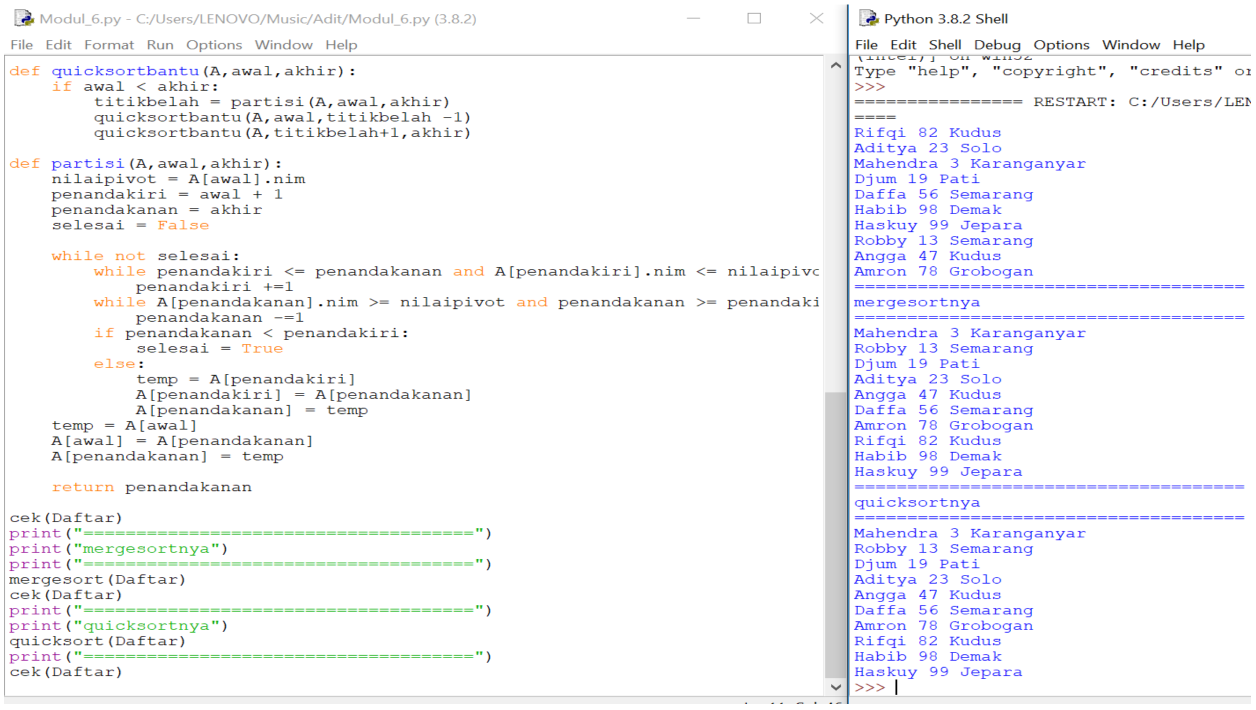
Nomor 1



```
from mahasiswa import *
c0 = mahasiswa('Rifqi', 82, 'Kudus', 200000)
c1 = mahasiswa('Aditya', 23, 'Solo', 300000)
c2 = mahasiswa('Mahendra', 3, 'Karanganyar', 400000)
c3 = mahasiswa('Djum', 19, 'Pati', 500000)
c4 = mahasiswa('Daffa', 56, 'Semarang', 600000)
c5 = mahasiswa('Habib', 98, 'Demak', 700000)
c6 = mahasiswa('Haskuy', 99, 'Jepara', 800000)
c7 = mahasiswa('Robby', 13, 'Semarang', 100000)
c8 = mahasiswa('Angga', 47, 'Kudus', 450000)
c9 = mahasiswa('Amron', 78, 'Grobogan', 650000)

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
```

Ln: 11 Col: 46

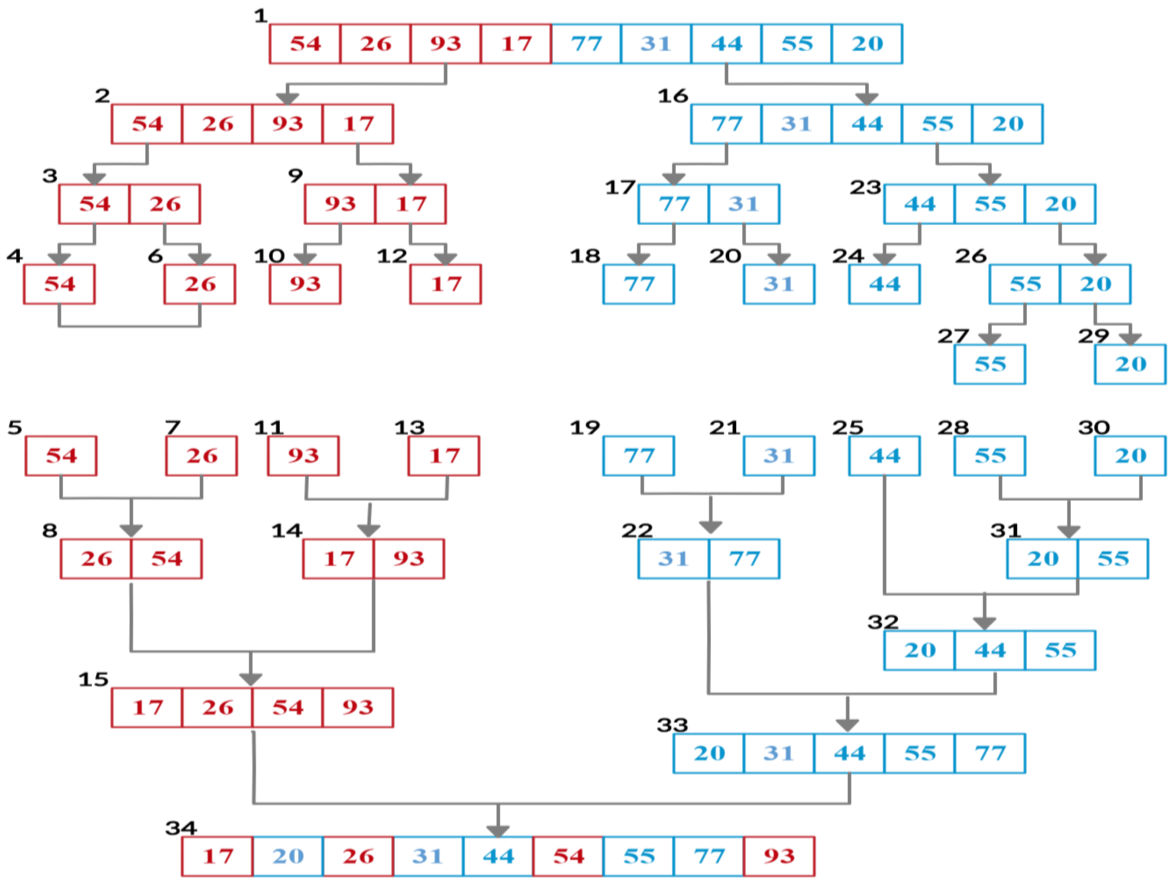
```
(Intel)] on win32
Type "help", "copyright", "credits" o
>>>
================ RESTART: C:/Users/LEN
====
Rifqi 82 Kudus
Aditya 23 Solo
Mahendra 3 Karanganyar
Djum 19 Pati
Daffa 56 Semarang
Habib 98 Demak
Haskuy 99 Jepara
Robby 13 Semarang
Angga 47 Kudus
Amron 78 Grobogan
=====================================
mergesortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
=====================================
quicksortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
>>>
```



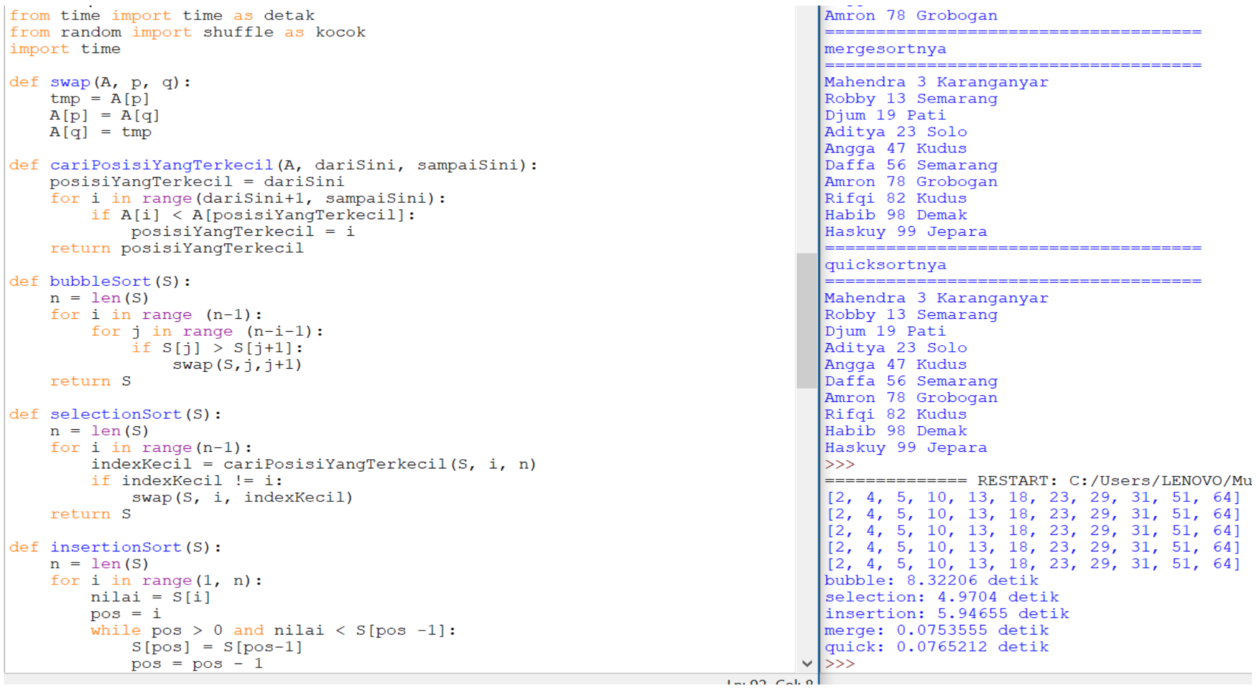
```
def quicksortbantu(A,awal,akhir):
    if awal < akhir:
        titikbelah = partisi(A,awal,akhir)
        quicksortbantu(A,awal,titikbelah -1)
        quicksortbantu(A,titikbelah+1,akhir)

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 <= nilaipivc
            penandakiri +=1
        while A[penandakanan].nim >= nilaipivot and penandakanan >= penandaki
            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

cek(Daftar)
print("=====================================")
print("mergesortnya")
print("=====================================")
mergesort(Daftar)
cek(Daftar)
print("=====================================")
print("quicksortnya")
quicksort(Daftar)
print("=====================================")
cek(Daftar)
```

```
(Intel)] on win32
Type "help", "copyright", "credits" o
>>>
================ RESTART: C:/Users/LEN
====
Rifqi 82 Kudus
Aditya 23 Solo
Mahendra 3 Karanganyar
Djum 19 Pati
Daffa 56 Semarang
Habib 98 Demak
Haskuy 99 Jepara
Robby 13 Semarang
Angga 47 Kudus
Amron 78 Grobogan
=====================================
mergesortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
=====================================
quicksortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
>>>
```

Nomor 2



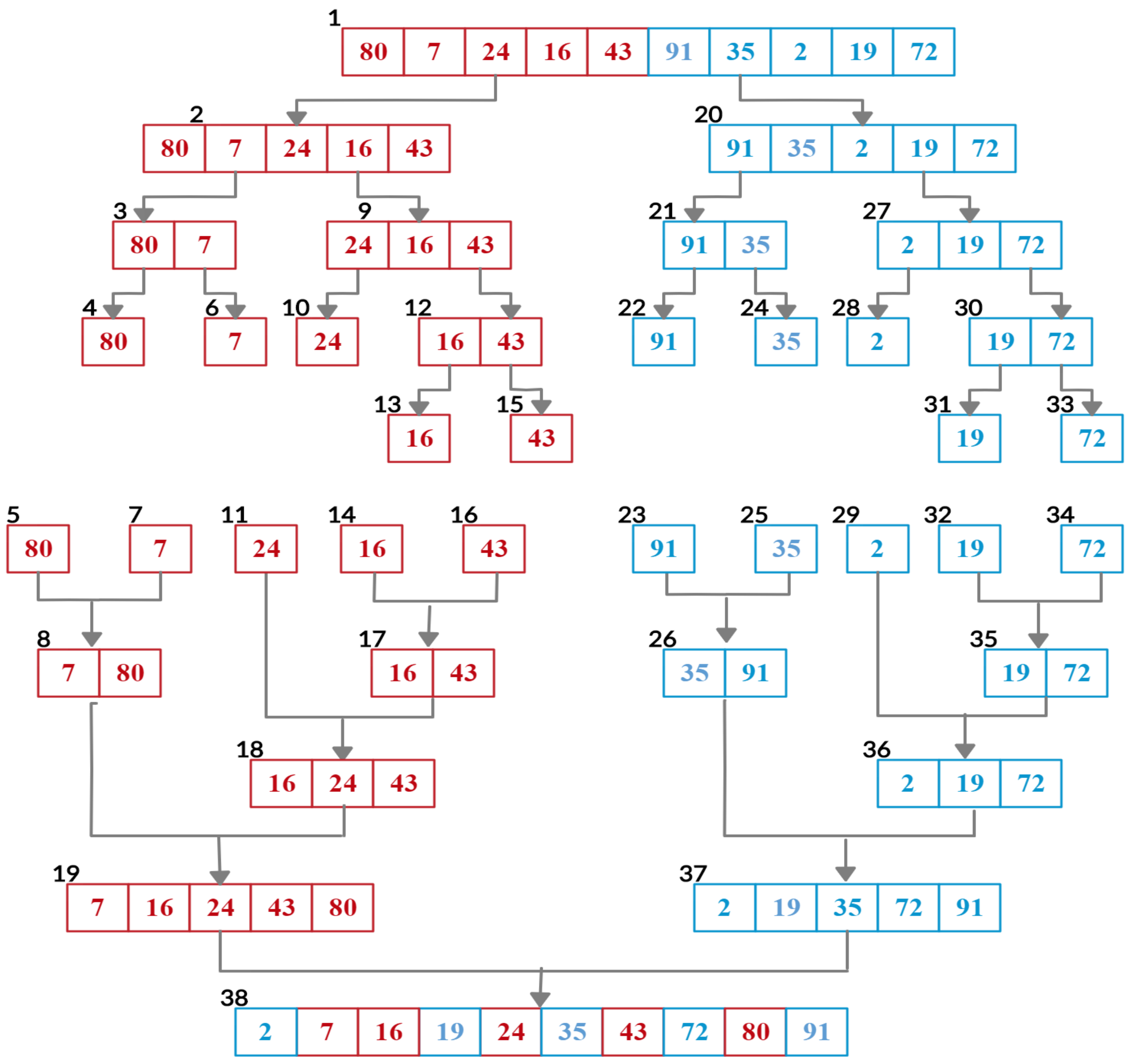
Nomor 3



```
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
```

```
Amron 78 Grobogan
=====================================
mergesortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
=====================================
quicksortnya
=====================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
>>>
============== RESTART: C:/Users/LENOVO/Mu
[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: 8.32206 detik
selection: 4.9704 detik
insertion: 5.94655 detik
merge: 0.0753555 detik
quick: 0.0765212 detik
>>>
```

Nomor 4a



1
80 7 24 16 43 91 35 2 19 72
2
80 7 24 16 43
20
91 35 2 19 72
3
80 7
9
24 16 43
21
91 35
27
2 19 72
4
80
6
7
10
24
12
16 43
22
91
24
35
28
2
30
19 72
13
16
15
43
31
19
33
72
5
80
7
7
11
24
14
16
16
43
23
91
25
35
29
2
32
19
34
72
8
7 80
17
16 43
26
35 91
35
19 72
18
16 24 43
36
2 19 72
19
7 16 24 43 80
37
2 19 35 72 91
38
2 7 16 19 24 35 43 72 80 91

Nomor 4b

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				

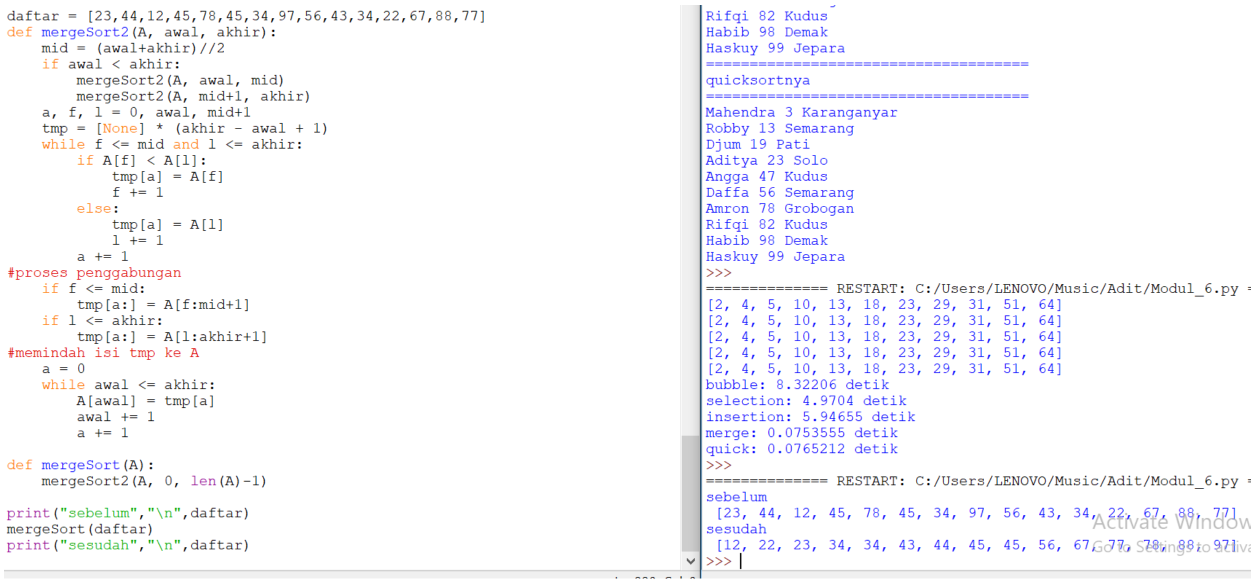
				pivot					
2	7	19	16	24	43	35	72	80	91
				Low	High				

		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

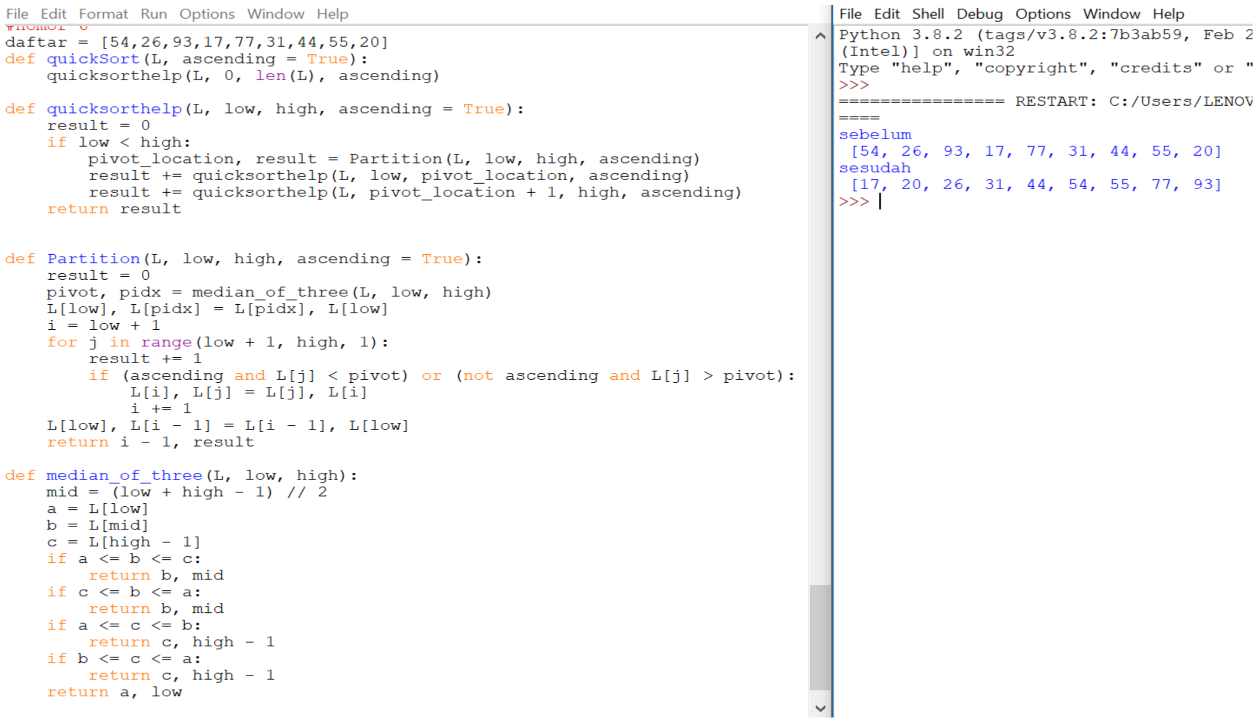
Nomor 5



```python
daftar = [23,44,12,45,78,45,34,97,56,43,34,22,67,88,77]
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

def mergeSort(A):
    mergeSort2(A, 0, len(A)-1)

print("sebelum","\n",daftar)
mergeSort(daftar)
print("sesudah","\n",daftar)
```

```
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
========================================
quicksortnya
========================================
Mahendra 3 Karanganyar
Robby 13 Semarang
Djum 19 Pati
Aditya 23 Solo
Angga 47 Kudus
Daffa 56 Semarang
Amron 78 Grobogan
Rifqi 82 Kudus
Habib 98 Demak
Haskuy 99 Jepara
>>>
============== RESTART: C:/Users/LENOVO/Music/Adit/Modul_6.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: 8.32206 detik
selection: 4.9704 detik
insertion: 5.94655 detik
merge: 0.0753555 detik
quick: 0.0765212 detik
>>>
============== RESTART: C:/Users/LENOVO/Music/Adit/Modul_6.py =
sebelum
 [23, 44, 12, 45, 78, 45, 34, 97, 56, 43, 34, 22, 67, 88, 77]
sesudah
 [12, 22, 23, 34, 34, 43, 44, 45, 45, 56, 67, 77, 78, 88, 97]
>>>
```

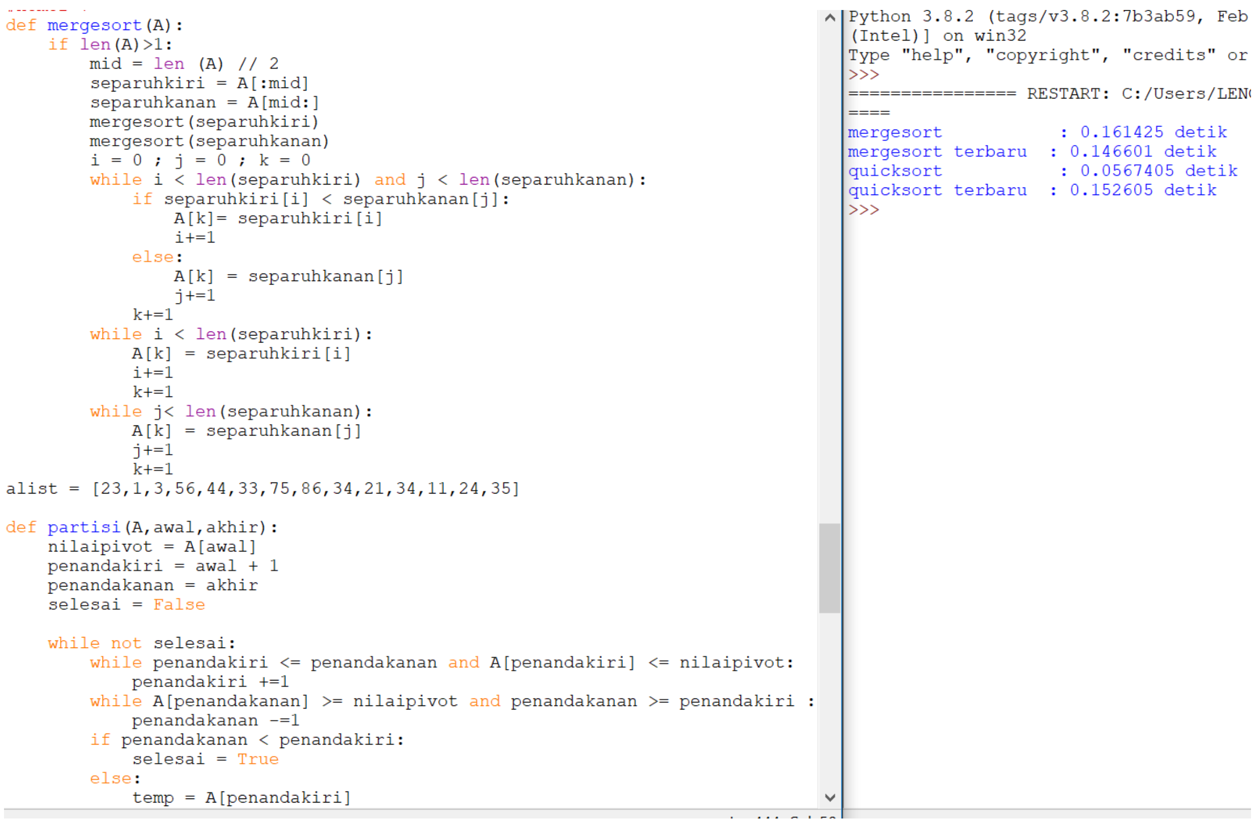
Nomor 6



```python
#nomor 6
daftar = [54,26,93,17,77,31,44,55,20]
def quickSort(L, ascending = True):
    quicksorthelp(L, 0, len(L), 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, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], 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
```

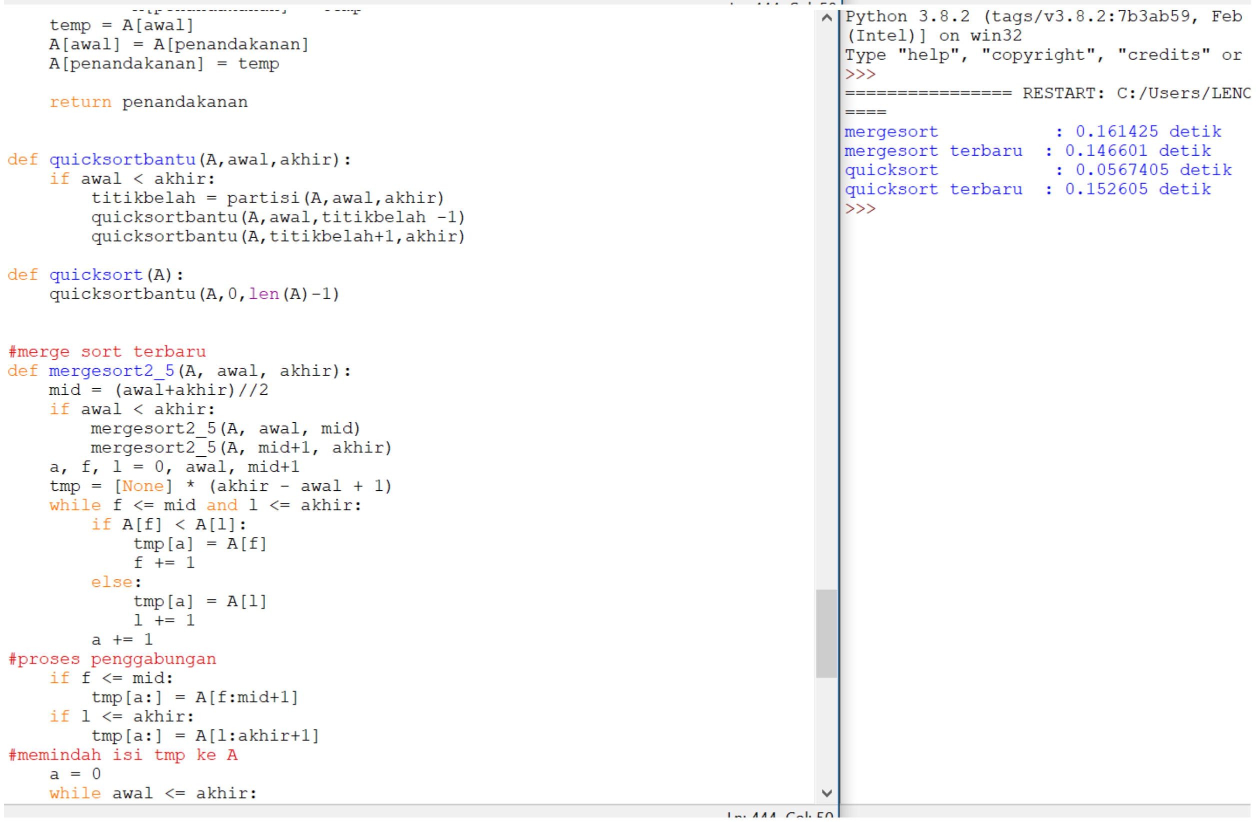
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 2
(Intel)] on win32
Type "help", "copyright", "credits" or "
>>>
================ RESTART: C:/Users/LENOV
====
sebelum
 [54, 26, 93, 17, 77, 31, 44, 55, 20]
sesudah
 [17, 20, 26, 31, 44, 54, 55, 77, 93]
>>>
```

Nomor 7



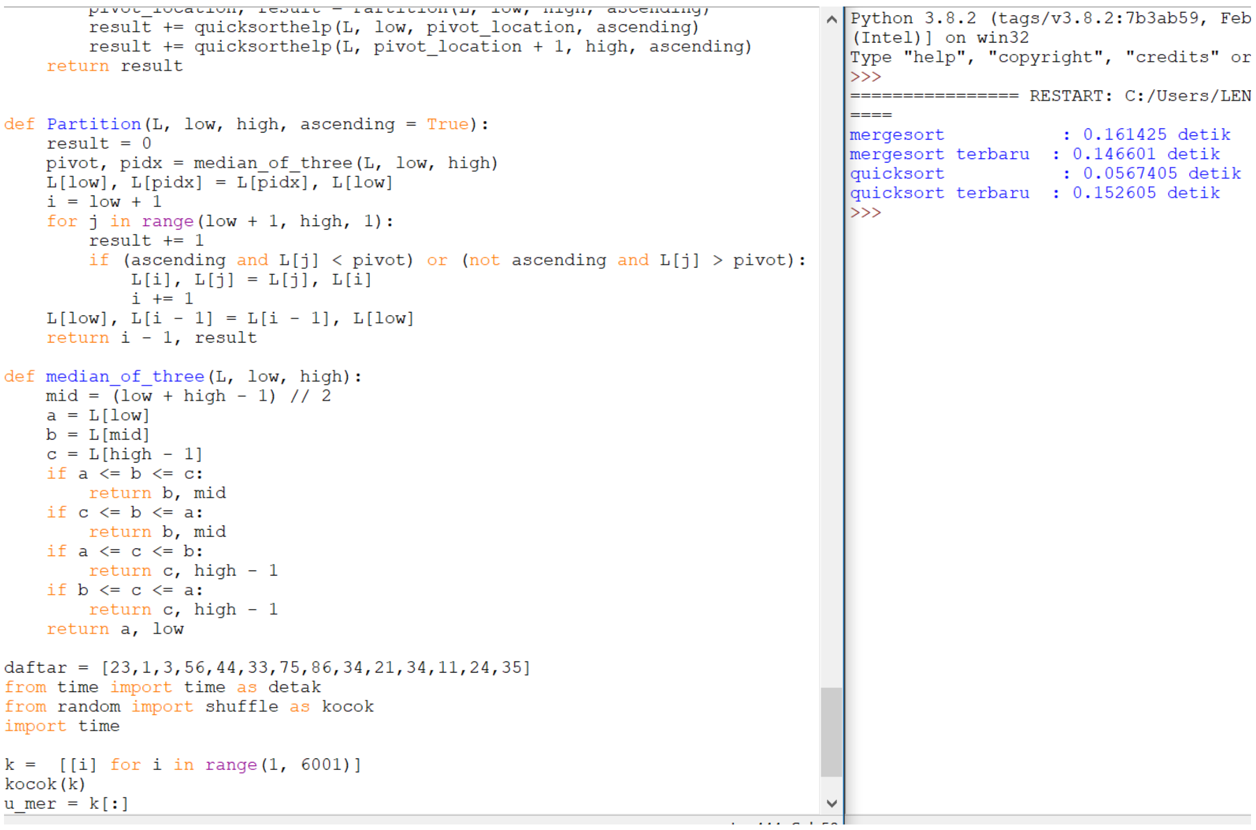
```
def mergesort(A):
    if len(A)>1:
        mid = len (A) // 2
        separuhkiri = A[:mid]
        separuhkanan = A[mid:]
        mergesort(separuhkiri)
        mergesort(separuhkanan)
        i = 0 ; j = 0 ; k = 0
        while i < len(separuhkiri) and j < len(separuhkanan):
            if separuhkiri[i] < separuhkanan[j]:
                A[k]= separuhkiri[i]
                i+=1
            else:
                A[k] = separuhkanan[j]
                j+=1
            k+=1
        while i < len(separuhkiri):
            A[k] = separuhkiri[i]
            i+=1
            k+=1
        while j< len(separuhkanan):
            A[k] = separuhkanan[j]
            j+=1
            k+=1
alist = [23,1,3,56,44,33,75,86,34,21,34,11,24,35]

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 +=1
        while A[penandakanan] >= nilaipivot and penandakanan >= penandakiri :
            penandakanan -=1
        if penandakanan < penandakiri:
            selesai = True
        else:
            temp = A[penandakiri]
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
(Intel)] on win32
Type "help", "copyright", "credits" or
>>>
================ RESTART: C:/Users/LEN
====
mergesort           : 0.161425 detik
mergesort terbaru  : 0.146601 detik
quicksort           : 0.0567405 detik
quicksort terbaru  : 0.152605 detik
>>>
```



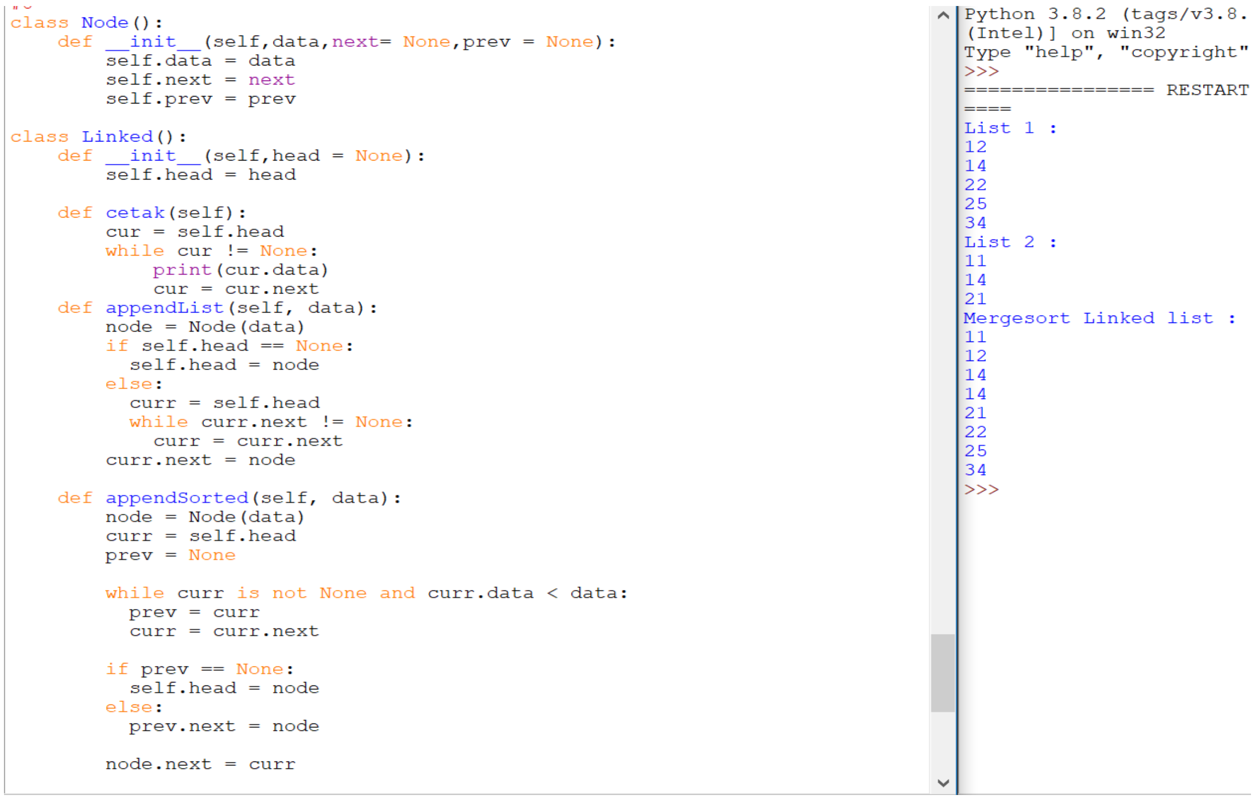
```
    temp = A[awal]
    A[awal] = A[penandakanan]
    A[penandakanan] = temp

    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)


#merge sort terbaru
def mergesort2_5(A, awal, akhir):
    mid = (awal+akhir)//2
    if awal < akhir:
        mergesort2_5(A, awal, mid)
        mergesort2_5(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:
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
(Intel)] on win32
Type "help", "copyright", "credits" or
>>>
================ RESTART: C:/Users/LENC
====
mergesort           : 0.161425 detik
mergesort terbaru  : 0.146601 detik
quicksort           : 0.0567405 detik
quicksort terbaru  : 0.152605 detik
>>>
```

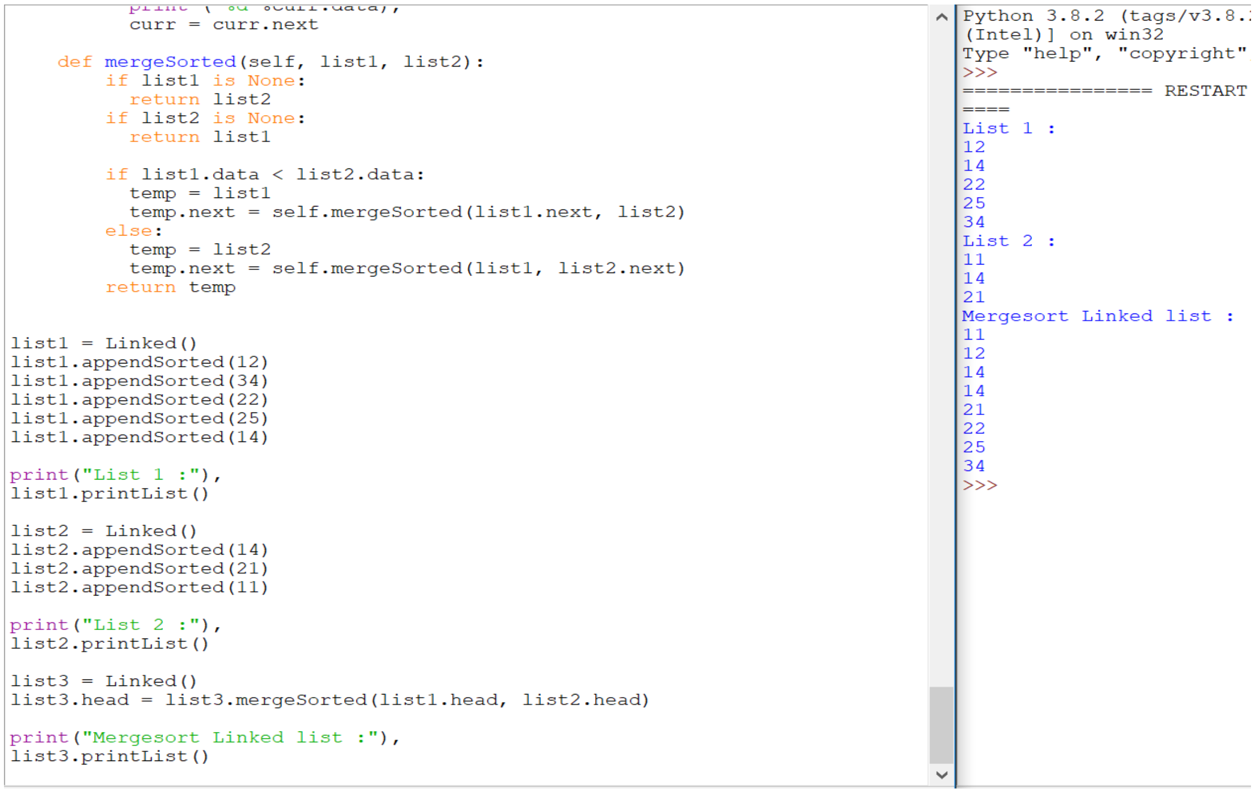

```
        pivot_location, result = Partition(L, low, high, ascending)
        result += quicksorthelp(L, low, pivot_location, ascending)
        result += quicksorthelp(L, pivot_location + 1, high, ascending)
    return result


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

daftar = [23,1,3,56,44,33,75,86,34,21,34,11,24,35]
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[:]
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
(Intel)] on win32
Type "help", "copyright", "credits" or
>>>
================ RESTART: C:/Users/LEN
====
mergesort           : 0.161425 detik
mergesort terbaru  : 0.146601 detik
quicksort           : 0.0567405 detik
quicksort terbaru  : 0.152605 detik
>>>
```

Nomor 8


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

class Linked():
    def __init__(self,head = None):
        self.head = head

    def cetak(self):
        cur = self.head
        while cur != None:
            print(cur.data)
            cur = cur.next
    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
```

```
Python 3.8.2 (tags/v3.8.
(Intel)] on win32
Type "help", "copyright"
>>>
================ RESTART
====
List 1 :
12
14
22
25
34
List 2 :
11
14
21
Mergesort Linked list :
11
12
14
14
21
22
25
34
>>>
```



```
          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(12)
list1.appendSorted(34)
list1.appendSorted(22)
list1.appendSorted(25)
list1.appendSorted(14)

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

list2 = Linked()
list2.appendSorted(14)
list2.appendSorted(21)
list2.appendSorted(11)

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

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

print("Mergesort Linked list :"),
list3.printList()
```

```
Python 3.8.2 (tags/v3.8.
(Intel)] on win32
Type "help", "copyright"
>>>
================ RESTART
====
List 1 :
12
14
22
25
34
List 2 :
11
14
21
Mergesort Linked list :
11
12
14
14
21
22
25
34
>>>
```