**LATIHAN DAN TUGAS PRAKTIKUM**

**MODUL 6**

**PRAKTIKUM ALGORITMA DAN STRUKTUR DATA**



DISUSUN OLEH:

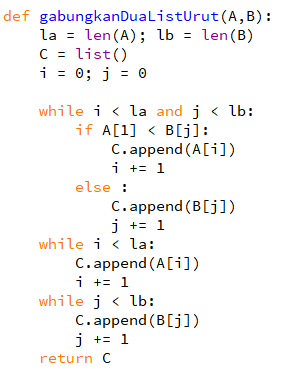
Nama : Syifaul Qolbi Auliya’ Darojat

NIM : L200200141

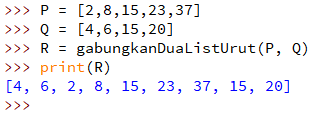
KELAS : F

**UNIVERSITAS MUHAMMADIYAH SURAKARTA**

Latihan 1



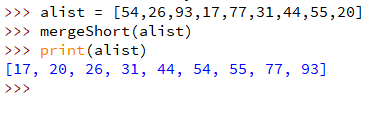
Hasil :



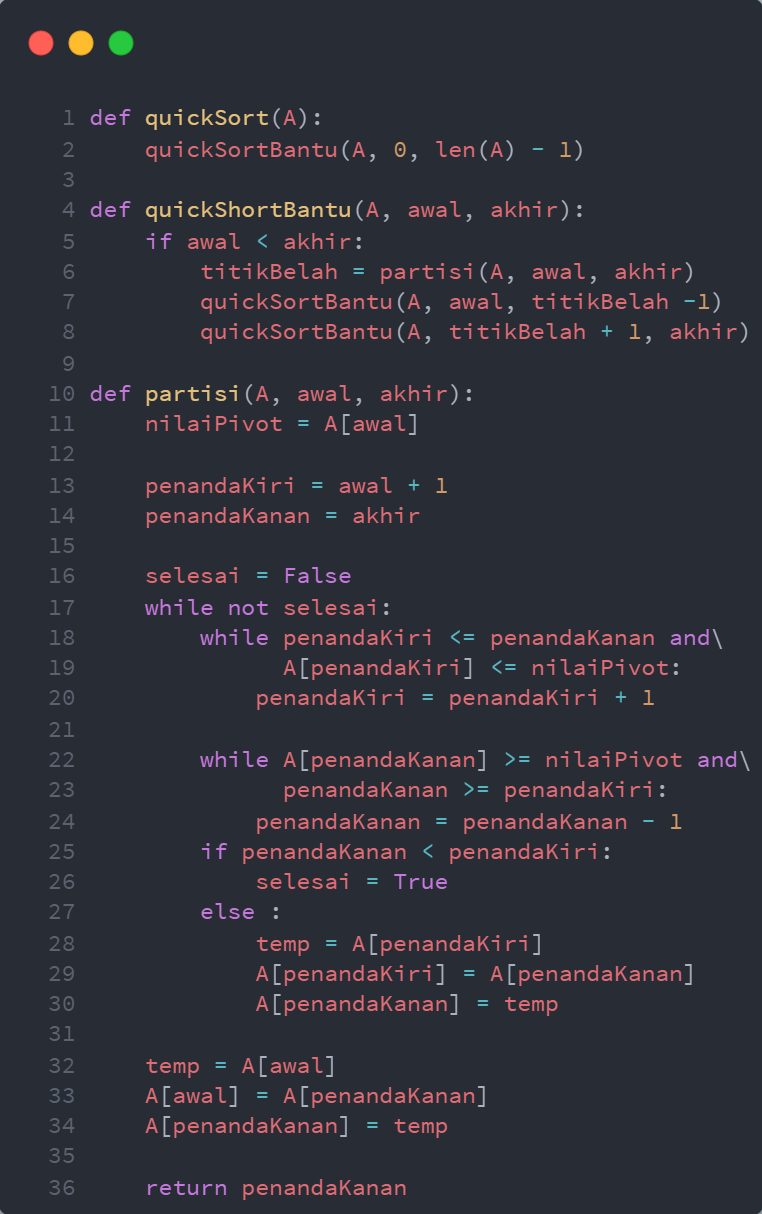
Latihan 2



Hasil :



Latihan 3



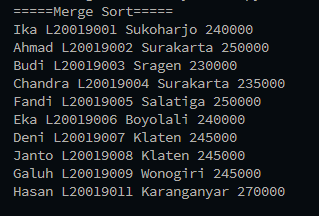
Soal Tugas :

**Nomer 1**

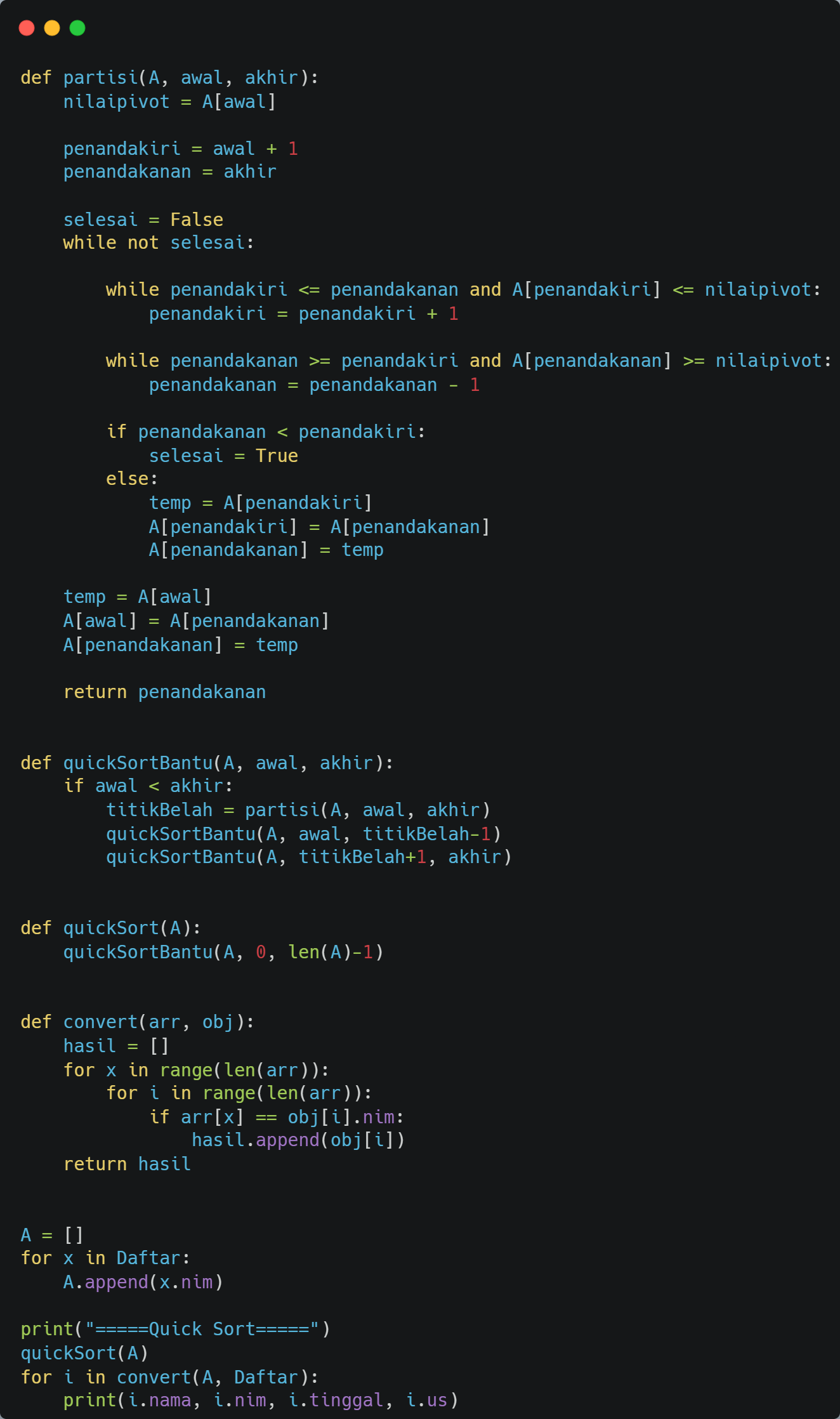


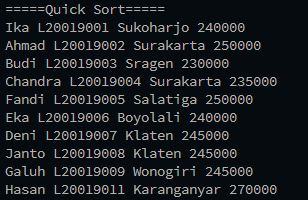
Mergeshort





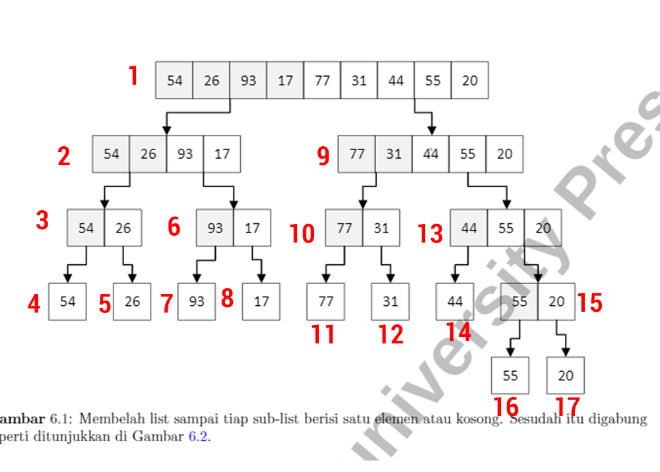
Quickshort



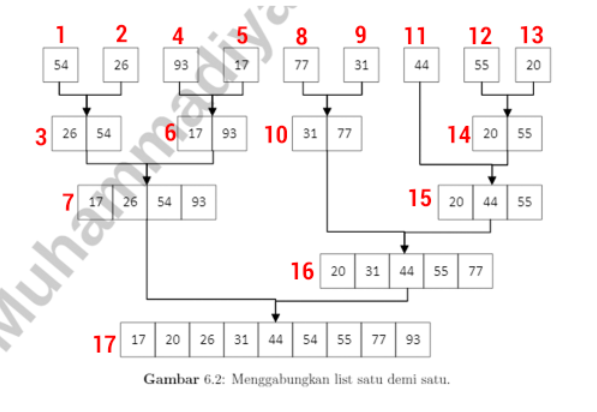


**Nomer 2**

Mergeshort



Quickshort



**Nomer 3**

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):

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

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)

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))

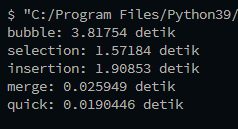
aw = detak()

quickSort(u\_qck)

ak = detak()

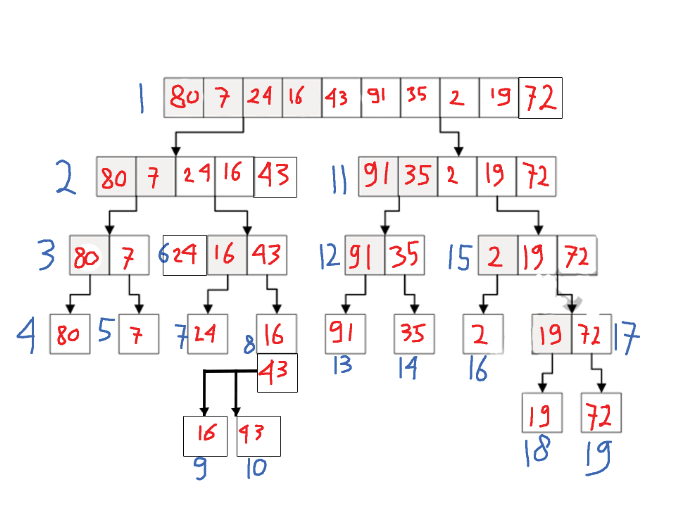
print("quick: %g detik" % (ak-aw))

Hasil :

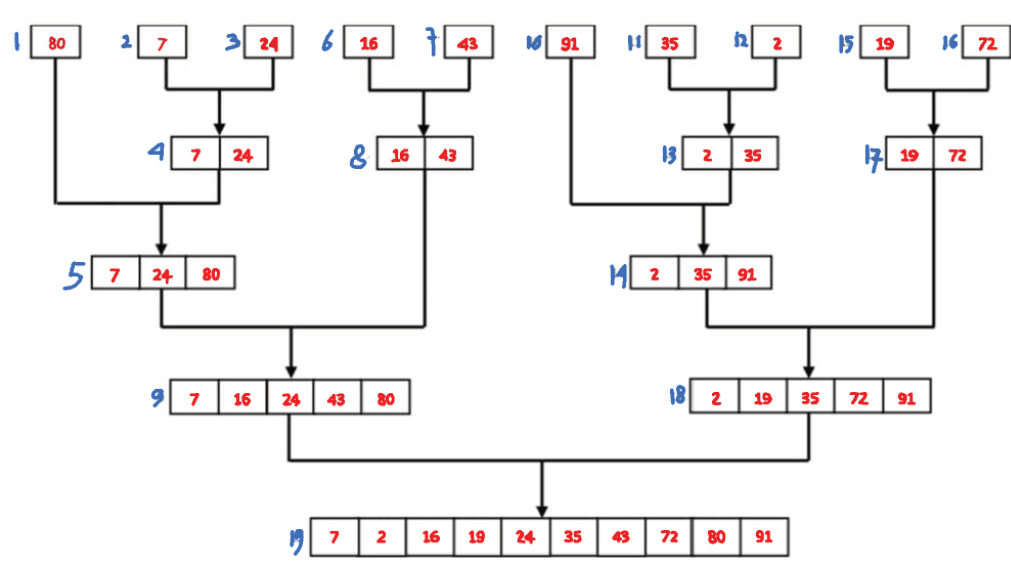


**Nomer 4**

Mergeshort



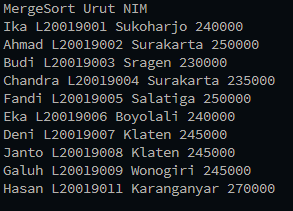
Quickshort



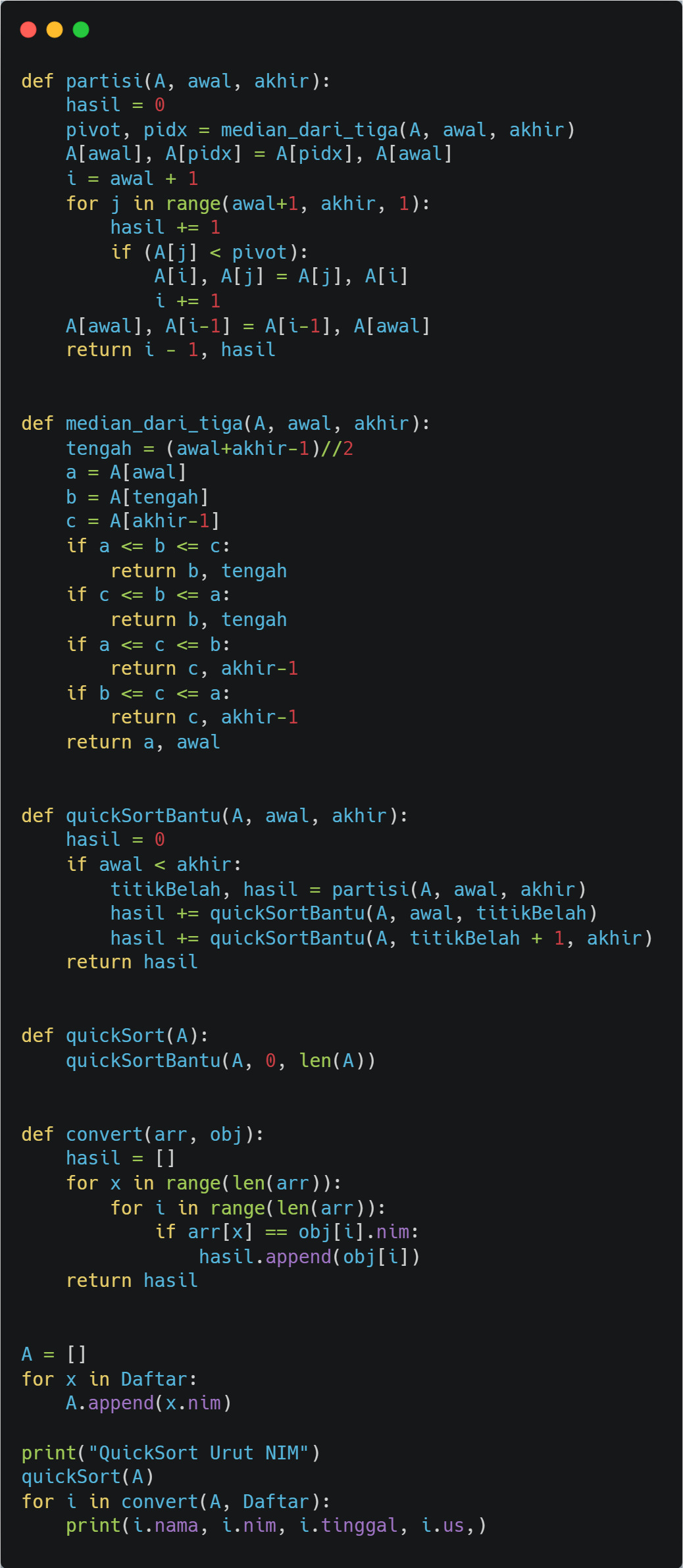
**Nomer 5**

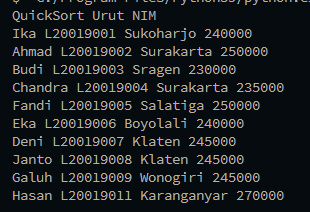


Hasil :



**Nomer 6**





**Nomer 7**

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 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 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

if f <= mid:

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

if l <= akhir:

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

a = 0

while awal <= akhir:

A[awal] = tmp[a]

awal += 1

a += 1

def mergeSortNew(A):

mergeSort2(A, 0, len(A)-1)

def quickSortNew(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 = quickSortNew(kurang)

lebih = quickSortNew(lebih)

return kurang + pivotList + lebih

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

print(daftar)

mergeSort(daftar)

quickSort(daftar)

mergeSortNew(daftar)

quickSortNew(daftar)

k = [[i] for i in range(1, 6001)]

kocok(k)

u\_mrg = k[:]

u\_qck = k[:]

u\_mrgNew = k[:]

u\_qckNew = k[:]

aw = detak()

mergeSort(u\_mrg)

ak = detak()

print("Merge v.1: %g detik" % (ak-aw))

aw = detak()

quickSort(u\_qck)

ak = detak()

print("Quick v.1: %g detik" % (ak-aw))

aw = detak()

mergeSortNew(u\_mrgNew)

ak = detak()

print("Merge v.2: %g detik" % (ak-aw))

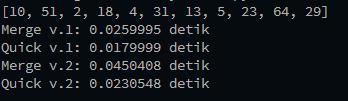
aw = detak()

quickSortNew(u\_qckNew)

ak = detak()

print("Quick v.2: %g detik" % (ak-aw))

Hasil :



**Nomer 8**

class Node():

def \_\_init\_\_(self, data, tautan=None):

self.data = data

self.tautan = tautan

def cetak(head):

curr = head

while curr is not None:

try:

print(curr.data)

curr = curr.tautan

except:

pass

a = Node(80)

b = Node(7)

c = Node(24)

d = Node(16)

e = Node(43)

f = Node(91)

g = Node(35)

h = Node(2)

i = Node(19)

j = Node(72)

a.tautan = b

b.tautan = c

c.tautan = d

d.tautan = e

e.tautan = f

f.tautan = g

g.tautan = h

h.tautan = i

i.tautan = j

def mergeSortLinkedList(A):

linked = A

try:

daftar = []

curr = A

while curr:

daftar.append(curr.data)

curr = curr.tautan

A = daftar

except:

A = A

if len(A) > 1:

mid = len(A) // 2

separuhkiri = A[:mid]

separuhkanan = A[mid:]

mergeSortLinkedList(separuhkiri)

mergeSortLinkedList(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

for x in A:

try:

linked.data = x

linked = linked.tautan

except:

pass

mergeSortLinkedList(a)

cetak(a)

Hasil :

