

Nama : Reza Aristo Rifandi

NIM : L200180206

Kelas : H

MODUL 6 Praktikum Algoritma dan Sturktur Data

Nomor 1 Merge Sort

```
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
```

```

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):
    hasil = []
    for x in range(len(arr)):
        for i in range(len(obj)):
            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)

```

```

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07
1)] on win32
Type "copyright", "credits" or "license()" for more
>>>
===== RESTART: D:\Informatika\Modul ASD\Modul-6\
MERGE SORT
2
4
5
10
13
18
23
29
31
51
64
>>> |

```

Nomor 1 Quick Sort

```

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(obj)):
            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)

===== RESTART: D:\Informatika\Modul ASD\Modul-6\
QUICK SORT
2
4
5
10
13
18
23
29
31
51
64
>>> |

```

Nomor 2

6.2 Merge sort

59

```

1 def mergeSort(A):
2     #print("Membelah", A)
3     if len(A) > 1:
4         mid = len(A) // 2 # Membelah list.
5         separuhKiri = A[:mid] # Slicing ini langkah yang expensive sebenarnya.
6         separuhKanan = A[mid:] # bisakah kamu membuatnya lebih baik?
7
8         mergeSort(separuhKiri) # Ini rekursi. Memanggil lebih lanjut mergeSort
9         mergeSort(separuhKanan) # untuk separuhKiri dan separuhKanan.
10
11     # Di bawah ini adalah proses penggabungan.
12     i=0 ; j=0 ; k=0
13     while i < len(separuhKiri) and j < len(separuhKanan):
14         if separuhKiri[i] < separuhKanan[j]: # while-loop ini
15             A[k] = separuhKiri[i] # menggabungkan kedua list, yakni
16             i = i + 1 # separuhKiri dan separuhKanan,
17         else: # sampai salah satu kosong.
18             A[k] = separuhKanan[j] # Perhatikan kesamaan strukturnya
19             j = j + 1 # dengan proses penggabungan
20             k=k+1 # dua list urut.
21
22     while i < len(separuhKiri): # Jika separuhKiri mempunyai sisa
23         A[k] = separuhKiri[i] # tumpukkan ke A
24         i = i + 1 # satu demi satu.
25         k = k + 1 #
26
27     while j < len(separuhKanan): # Jika separuhKanan mempunyai sisa
28         A[k] = separuhKanan[j] # tumpukkan ke A
29         j = j + 1 # satu demi satu.
30         k = k + 1 #
31     #print("Menggabungkan", A)

```

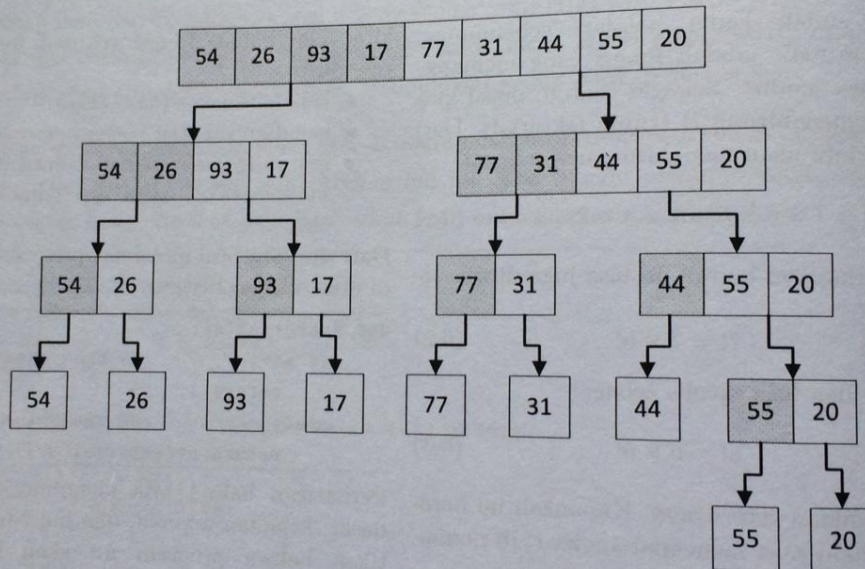
Larikan program di atas dengan memanggilnya seperti ini

```

alist = [54,26,93,17,77,31,44,55,20] *
mergeSort(alist) *
print(alist)

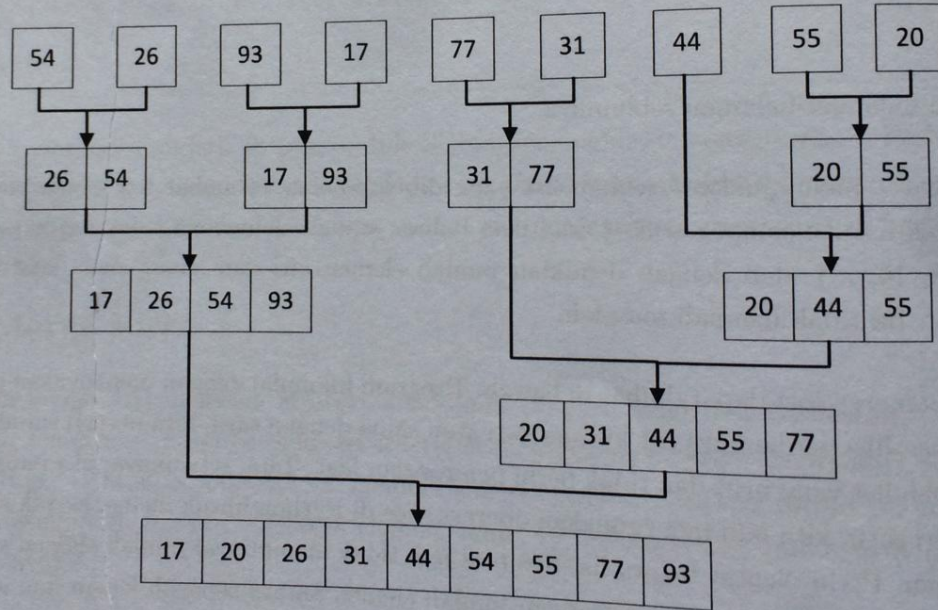
```

6.1 *



Gambar 6.1: Membelah list sampai tiap sub-list berisi satu elemen atau kosong. Sesudah itu digabung seperti ditunjukkan di Gambar 6.2.

6.2]



Gambar 6.2: Menggabungkan list satu demi satu.

Nomor 3

```
from time import time as detik
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

```



```

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)

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[:]

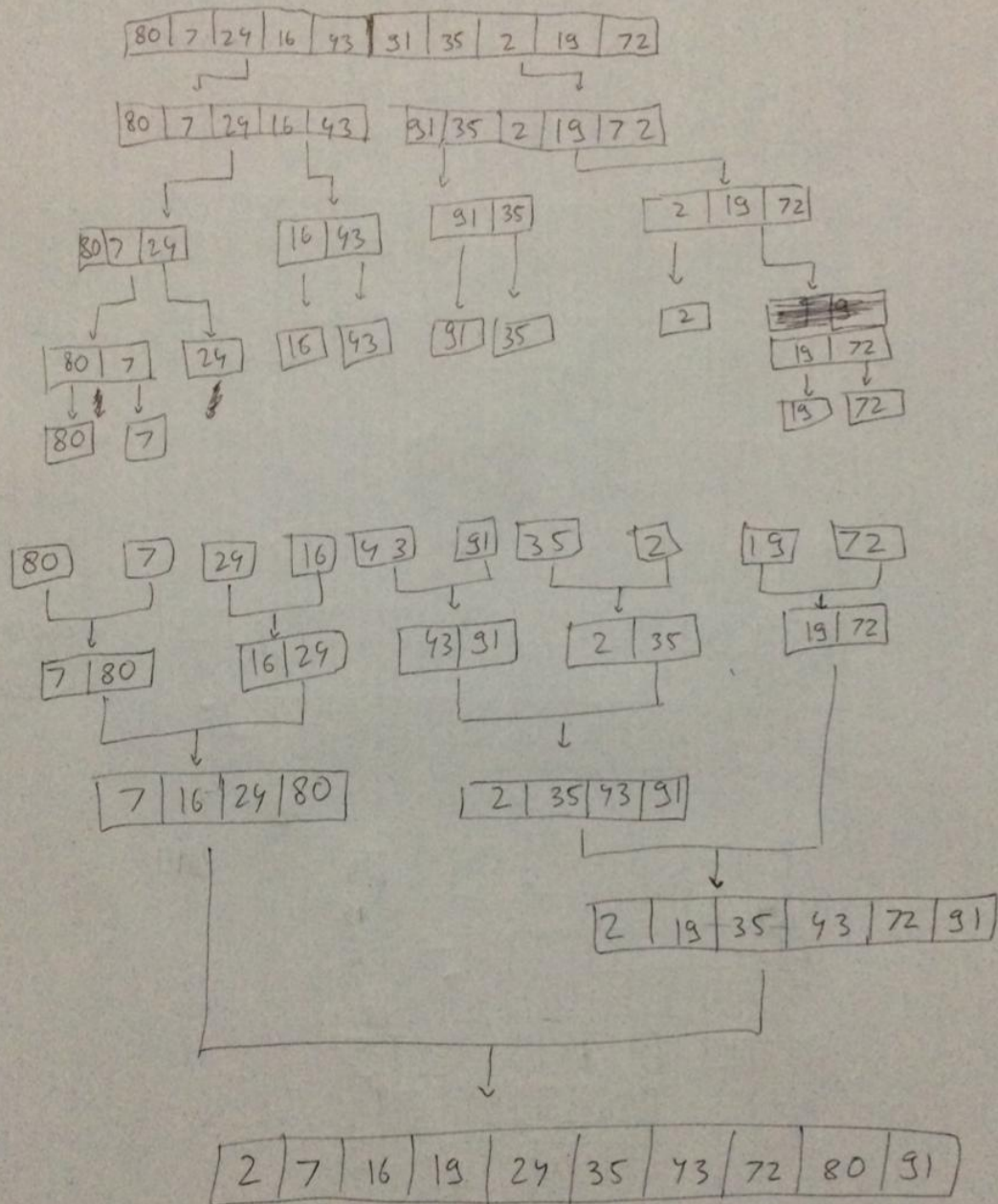
aw=detak();bubbleSort(u_bub);ak=detak();print("bubble: %g detik" %(ak-aw));
aw=detak();selectionSort(u_sel);ak=detak();print("selection: %g detik" %(ak-aw));
aw=detak();insertionSort(u_ins);ak=detak();print("insertion: %g detik" %(ak-aw));
aw=detak();mergeSort(u_mrg);ak=detak();print("merge: %g detik"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

```

Nomor 4

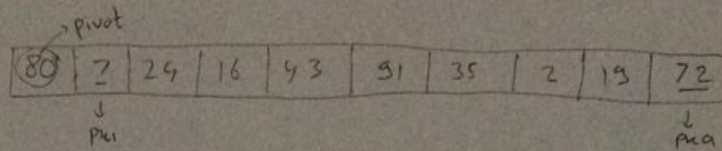
$L = [80, 7, 29, 16, 43, 31, 35, 2, 19, 72]$

a) Merge Sort

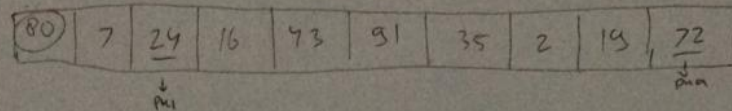


$L = [80, 7, 29, 16, 43, 91, 35, 2, 19, 72]$

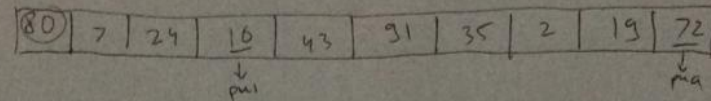
b) Quick Sort



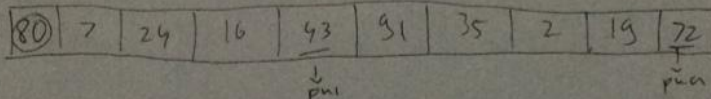
80 sebagai pivot
p_{hi} dan p_{ra}
akan berumpuk
pada titik tengah



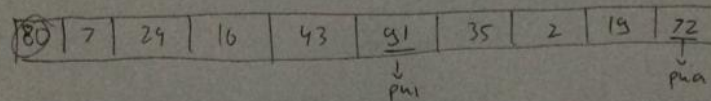
$7 < 80$ p_{hi} pindah ke depan



$24 < 80$ p_{hi} ke belakang



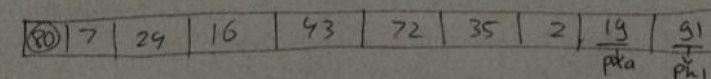
$16 < 80$ p_{hi} ke belakang



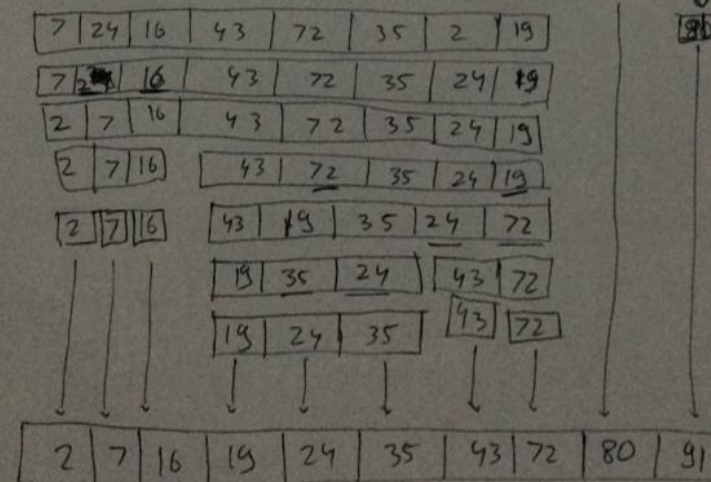
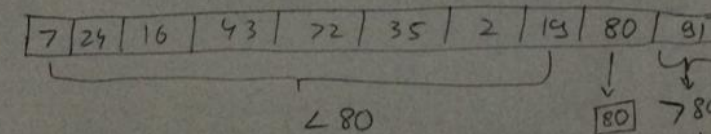
$43 < 80$ p_{hi} ke belakang
 $91 > 80$ p_{hi} berhenti
 $72 < 80$ p_{ra} berhenti



tukar posisi 72 dan 91



$19 < 80$ p_{ra} berhenti
 $91 > 80$ p_{hi} berhenti
titik tengah ditukarkan



Nomor 5

```
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
    while awal <= akhir:
        A[awal] = tmp[a]
        awal += 1
        a += 1

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

```

Nomor 6

```

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()
print("\nSetelah diurutkan")
quickSort(A)
cetak()

```


Nomor 7

```

from time import time as detik
from random import shuffle as kocok
import time

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]

mergeSort(daftar)
print (daftar)
quickSort(daftar)
print (daftar)
mergeSortNew(daftar)
print (daftar)
quickSortNew(daftar)
print (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: %g detik" %(ak-aw));
aw=detak();quickSort(u_qck);ak=detak();print("quick: %g detik" %(ak-aw));
aw=detak();mergeSortNew(u_mrgNew);ak=detak();print("merge New: %g detik" %(ak-aw));
aw=detak();quickSortNew(u_qckNew);ak=detak();print("quick New: %g detik" %(ak-aw));

```

Nomor 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(1)
b = Node(3)
c = Node(5)
d = Node(7)
e = Node(2)
f = Node(4)
g = Node(6)

a.tautan = b
b.tautan = c
c.tautan = d
d.tautan = e
e.tautan = f
f.tautan = g

def mergeSortLL(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:]

        mergeSortLL(separuhkiri)
        mergeSortLL(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

mergeSortLL(a)
cetak(a)

```