Aji Mustaqim

L200180141/E

ALGOSTRUK

LATIHAN

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04
:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for m
ore information.
>>>
============ RESTART: C:\Users\AJI TOSKA\Downloa
ds\algos\anu1.py ============
>>> print(C)
[2, 4, 6, 8, 15, 15, 20, 23, 37]
>>> |
```

```python
def gabungkanDuaListUrut(A, B):
    la = len(A); lb = len(B)
    C = list()
    i = 0; j = 0

    while i < la and j < lb:
        if A[i] < B[j]:
            C.append(A[i])
            i += 1
        else:
            C.append(B[j])
            j += 1

    while i < la:
        C.append(A[i])
        i += 1
    while j < lb:
        C.append(B[j])
        j += 1
    return C

A = [2, 8, 15, 23, 37]
B = [4, 6, 15, 20]
C = gabungkanDuaListUrut(A, B)
```

```
>>>
============ RESTART: C:/Users/AJI TOSKA/Music/algo/mod6/
aji2.py ============
>>> alist=[54,26,93,17,77,31,44,55,20]
>>> mergeSort(alist)
Membelah [54, 26, 93, 17, 77, 31, 44, 55, 20]
Membelah [54, 26, 93, 17]
Membelah [54, 26]
Membelah [54]
Menggabungkan [54]
Membelah [26]
Menggabungkan [26]
Menggabungkan [26, 54]
Membelah [93, 17]
Membelah [93]
Menggabungkan [93]
Membelah [17]
Menggabungkan [17]
Menggabungkan [17, 93]
Menggabungkan [17, 26, 54, 93]
Membelah [77, 31, 44, 55, 20]
Membelah [77, 31]
Membelah [77]
Menggabungkan [77]
Membelah [31]
Menggabungkan [31]
Menggabungkan [31, 77]
Membelah [44, 55, 20]
Membelah [44]
Menggabungkan [44]
Membelah [55, 20]
Membelah [55]
Menggabungkan [55]
Membelah [20]
Menggabungkan [20]
```

```python
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)
```

```python
def quickSort(A):
    quickSortBantu(A, 0, len(A) - 1)

A = [1,4,3,2,6,8,7]
quickSort(A)
print(quickSort)

def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah = partisi(A, awal, akhir)
        quickSortBantu(A, awal, titikBelah - 1)
        qucikSortBantu(A, titikBelah + 1, akhir)


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 <= nilaiPivot and\
              A[penandaKanan] <= penandaKiri:
            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
```

TUGAS

1

```python
print("\nNomor 1")
class Mahasiswa(object):
    def __init__(self, nama, NIM, kota, us):
        self.nama = nama
        self.NIM = NIM
        self.kotaTinggal = kota
        self.uangSaku = us

a0 = Mahasiswa('Ika', 10, 'Sukoharjo', 240000)
a1 = Mahasiswa('Budi', 51, 'Sragen', 230000)
a2 = Mahasiswa('Ahmad', 2, 'Surakarta', 250000)
a3 = Mahasiswa('Chandra', 18, 'Surakarta', 235000)
a4 = Mahasiswa('Eka', 4, 'Boyolali', 240000)
a5 = Mahasiswa('Fandi', 31, 'Salatiga', 250000)
a6 = Mahasiswa('Deni', 13, 'Klaten', 245000)
a7 = Mahasiswa('Galuh', 5, 'Wonogiri', 245000)
a8 = Mahasiswa('Janto', 23, 'Klaten', 245000)
a9 = Mahasiswa('Hasan', 64, 'Karanganyar', 270000)
a10 = Mahasiswa('Khalid', 29, 'Purwodadi', 230000)

Daftar = [a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]

def urutkanNIM(a):
    baru = {}
    for i in range(len(a)):
        baru[a[i].nama] = a[i].NIM
    listofTuples = sorted(baru.items(), key = lambda x: x[1])
    for elem in listofTuples:
        print (elem[0], ':', elem[1])

urutkanNIM(Daftar)
```

```
========= RESTART: C:/Users/AJI TOSKA/Music/algo/mod6/Modul6
tugas.py =========

Nomor 1
Ahmad : 2
Eka : 4
Galuh : 5
Ika : 10
Deni : 13
Chandra : 18
Janto : 23
Khalid : 29
Fandi : 31
Budi : 51
Hasan : 64
```

2

```python
print("\nNomor 2")
def bubblesort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

def gabung(a,b):
    c = []
    c = a+b
    n = len(c)
    for i in range(n):
        for j in range(0, n-i-1):
            if c[j] > c[j+1]:
                c[j], c[j+1] = c[j+1], c[j]
    return c

a = [5,45,12,32,6,10,2]
b = [26,8,20,14,40]
a,b = bubblesort(a),bubblesort(b)

print(a)
print(b)
print(gabung(a,b))
```

```
Nomor 2
[2, 5, 6, 10, 12, 32, 45]
[8, 14, 20, 26, 40]
[2, 5, 6, 8, 10, 12, 14, 20, 26, 32, 40, 45]
```

3,4

```python
print("\nNomor 3 dan 4")
from time import time as detak
from random import shuffle as kocok
import time

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

def bubb(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]

def sele(A):
    for i in range(len(A)):
        min_idx = i
        for j in range(i+1, len(A)):
            if A[min_idx] > A[j]:
                min_idx = j
        A[i], A[min_idx] = A[min_idx], A[i]

def inse(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
```

```python
            arr[j+1] = key

def mergeSort(arr):
    if len(arr) >1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+=1
            else:
                arr[k] = R[j]
                j+=1
            k+=1
        while i < len(L):
            arr[k] = L[i]
            i+=1
            k+=1
        while j < len(R):
            arr[k] = R[j]
            j+=1
            k+=1

def partition(arr,low,high):
    i = ( low-1 )
    pivot = arr[high]
    for j in range(low , high):
        if   arr[j] <= pivot:
            i = i+1
            arr[i],arr[j] = arr[j],arr[i]
    arr[i+1],arr[high] = arr[high],arr[i+1]
    return ( i+1 )

def quickSort(arr,low,high):
    if low < high:
        pi = partition(arr,low,high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

bub = k[:]
sel = k[:]
ins = k[:]
mer = k[:]
qui = k[:]

aw=detak();bubb(bub);ak=detak();print('bubble : %g detik' %(ak-aw));
aw=detak();sele(sel);ak=detak();print('selection : %g detik' %(ak-aw));
aw=detak();inse(ins);ak=detak();print('insertion : %g detik' %(ak-aw));
aw=detak();mergeSort(mer);ak=detak();print('merge : %g detik' %(ak-aw));
aw=detak();quickSort(qui,0,len(qui)-1);ak=detak();print('quick : %g detik' %(ak-
```

```
Nomor 3 dan 4
bubble : 22.5303 detik
selection : 8.07421 detik
insertion : 10.8908 detik
merge : 0.175195 detik
quick : 0.0840752 detik
```

```python
print("\nNomor 5")
import random

def _merge_sort(indices, the_list):
    start = indices[0]
    end = indices[1]
    half_way = (end - start)//2 + start
    if start < half_way:
        _merge_sort((start, half_way), the_list)
    if half_way + 1 <= end and end - start != 1:
        _merge_sort((half_way + 1, end), the_list)

    sort_sub_list(the_list, indices[0], indices[1])
    return the_list

def sort_sub_list(the_list, start, end):
    orig_start = start
    initial_start_second_list = (end - start)//2 + start + 1
    list2_first_index = initial_start_second_list
    new_list = []
    while start < initial_start_second_list and list2_first_index <= end:
        first1 = the_list[start]
        first2 = the_list[list2_first_index]
        if first1 > first2:
            new_list.append(first2)
            list2_first_index += 1
        else:
            new_list.append(first1)
            start += 1
    while start < initial_start_second_list:
        new_list.append(the_list[start])
        start += 1
    while list2_first_index <= end:
        new_list.append(the_list[list2_first_index])
        list2_first_index += 1
    for i in new_list:
        the_list[orig_start] = i
        orig_start += 1
    return the_list

def merge_sort(the_list):
    return _merge_sort((0, len(the_list) - 1), the_list)

print(merge_sort([13,45,12]))

Nomor 5
[12, 13, 45]
```

```python
print("\nNomor 6")
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

liste1 = list([12,4,15,124,123])
quickSort(liste1, False)  # descending order
print('sorted :', liste1)
```

```
Nomor 6
sorted : [124, 123, 15, 12, 4]
```

7

```python
print("\nNomor 7")
from time import time as detak
from random import shuffle as kocok
import time

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

def mergeSort(arr):
    if len(arr) >1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+=1
            else:
                arr[k] = R[j]
                j+=1
            k+=1
        while i < len(L):
            arr[k] = L[i]
            i+=1
            k+=1
        while j < len(R):
            arr[k] = R[j]
            j+=1
            k+=1
```

```python
def partition(arr,low,high):
    i = ( low-1 )
    pivot = arr[high]
    for j in range(low , high):
        if   arr[j] <= pivot:
            i = i+1
            arr[i],arr[j] = arr[j],arr[i]
    arr[i+1],arr[high] = arr[high],arr[i+1]
    return ( i+1 )

def quickSort(arr,low,high):
    if low < high:
        pi = partition(arr,low,high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

import random

def _merge_sort(indices, the_list):
    start = indices[0]
    end = indices[1]
    half_way = (end - start)//2 + start
    if start < half_way:
        _merge_sort((start, half_way), the_list)
    if half_way + 1 <= end and end - start != 1:
        _merge_sort((half_way + 1, end), the_list)

    sort_sub_list(the_list, indices[0], indices[1])

def sort_sub_list(the_list, start, end):
    orig_start = start
    initial_start_second_list = (end - start)//2 + start + 1
    list2_first_index = initial_start_second_list
    new_list = []
```

```python
    while start < initial_start_second_list and list2_first_index <= end:
        first1 = the_list[start]
        first2 = the_list[list2_first_index]
        if first1 > first2:
            new_list.append(first2)
            list2_first_index += 1

        else:
            new_list.append(first1)
            start += 1

    while start < initial_start_second_list:
        new_list.append(the_list[start])
        start += 1

    while list2_first_index <= end:
        new_list.append(the_list[list2_first_index])
        list2_first_index += 1
    for i in new_list:
        the_list[orig_start] = i
        orig_start += 1

def merge_sort(the_list):
    return _merge_sort((0, len(the_list) - 1), the_list)

def quickSortMOD(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)
```

```python
        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

mer = k[:]
qui = k[:]
mer2 = k[:]
qui2 = k[:]
aw=detak();mergeSort(mer);ak=detak();print('merge : %g detik' %(ak-aw));
aw=detak();quickSort(qui,0,len(qui)-1);ak=detak();print('quick : %g detik' %(ak-
aw=detak();merge_sort(mer2);print('merge mod : %g detik' %(ak-aw));
aw=detak();quickSortMOD(qui2, False);print('quick mod : %g detik' %(ak-aw));

Nomor 7
merge : 0.173095 detik
quick : 0.0920861 detik
merge mod : -0.0140095 detik
quick mod : -0.223202 detik
```

8

```python
print("\nNomor 8")
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None
    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

    def printList(self):
        curr = self.head
```

```python
        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 = LinkedList()
list1.appendSorted(13)
list1.appendSorted(12)
list1.appendSorted(3)
list1.appendSorted(16)
list1.appendSorted(7)

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

list2 = LinkedList()
list2.appendSorted(9)
list2.appendSorted(10)
list2.appendSorted(1)

print("List 2 :"),
list2.printList()
list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)

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

```
Nomor 8
List 1 :
3
7
12
13
16
List 2 :
1
9
10
Merged List :
1
3
7
9
10
12
13
16
...
```